1-1-2012

Techniques For Sensor-Integrated Robotic Systems: Raman Spectra Analysis, Image Guidance, And Kinematic Calibration

Luke Anthony Reisner
Wayne State University,

TECHNIQUES FOR SENSOR-INTEGRATED ROBOTIC SYSTEMS: RAMAN SPECTRA ANALYSIS, IMAGE GUIDANCE, AND KINEMATIC CALIBRATION

by

LUKE ANTHONY REISNER

DISSERTATION

Submitted to the Graduate School

of Wayne State University,

Detroit, Michigan,

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

2012

MAJOR: COMPUTER ENGINEERING

Approved by:

_____

Advisor Date

_____

_____

_____

DEDICATION

This dissertation is dedicated to all the ladies out there. And also to my parents.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Alex Cao, Dr. Rachel Kast, Dr. Brady King, Dr. Michael Klein, and Dr. Abhilash Pandya for their support in this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1:
Introduction and Motivation

1.1 Robotics and Sensor Technology

The ability to indirectly or remotely manipulate objects within an environment can be extremely useful. For example, a surgeon may need to operate on portions of patient's body that are difficult or unsafe to reach with hand-held tools. A soldier could be commanded to search for explosive devices in a hostile environment, or an astronaut may be required to repair a module on the outside of a spacecraft or even collect specimens on a different planet. In all of these cases, indirect manipulation is either desired or required for both the success of the operation and the safety of those involved.

Increasingly, robotic technology is being used to meet the demands of these kinds of challenging applications. A robot can perform precise operations with manipulators or end-effectors that replace a person's hands or tools. The movements of a robot can be fully under the control of a human, or it can act partially or completely autonomously. Moreover, modern communication technology can be exploited to remotely execute desired operations from the next room, a different country, or even another planet. These properties can be very beneficial in many applications, such as the previously stated examples. For instance, a soldier can control a robotic vehicle from a safe distance to search for explosives, or an astronaut can direct a robotic arm to work on the exterior of a spacecraft without being exposed to the hazards of outer space.

In addition to indirect manipulation, the ability to sense properties of the environment can be very valuable. For example, a surgeon may desire a way to distinguish healthy and cancerous tissue during a tumor removal operation. A solder could use a device that detects trace amounts of explosive materials, or an astronaut may need the ability to analyze the composition of

extraterrestrial soil samples. In many cases, like those above, the properties being detected often fall outside the capabilities of human senses.

The answer to these diverse needs can be found in the advancing field of sensor technology. Sensors are tools that detect specific physical properties and produce an output that can be interpreted by another device (such as a computer) or a person. There are many types of sensors based on various technologies that provide valuable information for particular applications. For instance, one sensor could be designed to identify biochemical markers associated with cancer, and another could be built to detect chemical compounds found in common explosives.

Finally, robotics and sensor technology can be combined to form an extremely powerful solution to difficult problems. Sensors are an ideal complement to robotics for several reasons. Most commonly, they supply the feedback necessary for robot joint control. In addition, they can provide information that would otherwise be unavailable due to indirect manipulation. For example, they can enable force feedback or visual imaging of a remote environment. The combination of the technologies is also beneficial for sensors. Analytical sensors generally need to be positioned very accurately to obtain good performance and ensure the correct location is sensed, and robots are typically adept at precise placement.

Sensor-integrated robotic systems are currently being used or developed for many applications, such as the following three examples. First, the recent Mars Science Laboratory mission (as well as the upcoming ExoMars mission) is employing a rover with robotics and numerous sensors to remotely study the planet Mars [1]. Second, research on the detection of land mines involves single- and multi-robot systems equipped with various sensing technologies [2]. Third, work is being done to add force and tactile sensing to surgical robots [3]. Although

many of these kinds of integrated systems are still under development, they have the potential to provide significant benefits in many different fields.

1.1.1 Surgical Robotics

One field that exemplifies the use of indirect manipulation is surgical robotics. In a typical surgical procedure, the doctor has to navigate a patient's body and manipulate structures with great care and precision. A surgical robot helps the doctor by manipulating the surgical tools and/or holding an optical instrument known as an endoscope. The doctor generally directs the robot via hand controllers. The main surgical robot in use today is the da Vinci (Intuitive Surgical, Sunnyvale, CA), which has four arms for controlling the tools and endoscope.

A surgical robot leverages the strengths of robotics to aid the doctor performing the procedure. A robot can move a tool with high accuracy or tirelessly hold an instrument in a fixed location. By processing the doctor's controller inputs, the robot's control system can remove tremors in the doctor's hands or translate large movements by the doctor into much smaller movements inside the patient. In addition, the robot's arms can enter the patient through tiny incisions, minimizing the damage to the patient.

Although not commonly done today, surgical robots are excellent candidates for advanced sensor integration. The use of force or tactile sensors promises to restore the haptic feedback that is lost due to indirect manipulation. There are also a number of tissue properties that could be analyzed to help the doctor treat the patient. For example, an ultrasound sensor could be used to locate masses, and some type of biochemical sensor could be used to identify abnormal tissue.

1.1.2 Raman Spectroscopy

Raman spectroscopy is a good example of a powerful sensing technique. It is an optical technique that utilizes the inelastic scattering of light by a substance. Essentially, a device

known as a Raman spectrometer shines a laser beam at a specimen and records the light that comes back. Based on the amount of light that is collected a various wavelengths, a Raman spectrum is constructed. This spectrum acts like a fingerprint that provides information about the molecular or chemical composition of the specimen. (Section 2.2 has more detailed information on Raman spectroscopy.)

The amount of information contained in a Raman spectrum and its relative ease of acquisition make Raman spectroscopy an excellent choice for many applications. For instance, it can distinguish healthy tissue from cancerous tissue [4-6], and it can be used to detect explosive materials [7]. Raman spectroscopy is even planned for use on Mars (as part of the European Space Agency's ExoMars mission) to analyze the geology and look for evidence of microorganisms [8].

A Raman spectrometer is suitable for integration with a robotic system. The emission and collection of light can occur through a fiber optic probe, which can be routed to the end of a mobile robotic arm. Typically, the diameter of the device's sampling region is less than 1 mm, and the probe must be held in position for many seconds while the light is collected. A robot would be able to accurately reach the desired region and keep the device steady for as long as needed. This is similar to existing Raman microscopes, which use a motorized platform to position an acquired sample underneath the emitted laser light. By placing a Raman probe at the end of a surgical robot or other sampling arm, it becomes possible to analyze many different kinds of specimens in their native environments.

1.1.3 Image Guidance

The usage of sensors and/or robotic systems can be augmented with image guidance. Image guidance involves a combination of imaging and tracking sensors that are used to indicate the

positions of objects of interest (tools, etc.) on images of an environment. The imaging techniques can be very advanced (e.g., magnetic resonance imaging) or relatively simple (e.g., a camera). Computer visualization techniques such as virtual and augmented reality can be used to display the sensed data as well as any other relevant information. Image guidance is most frequently discussed in the context of surgical procedures (called image-guided surgery), but it is also employed in other fields, such as military [9] and outer space [10] applications.

Image guidance can aid in the navigation of an environment. For example, when trying to maneuver a robotic manipulator with respect to a structure of interest, it may be helpful to see the position of the robot in a 3D model of the environment derived from imaging data. Similarly, a sensor could be more easily moved to a specimen of interest using the same 3D information.

The visualization techniques associated with image guidance can also assist in the understanding of data collected with a sensor. Any time new data is collected, the source of the data can be displayed in the model of the environment. Moreover, any information derived from the sensor data can be integrated with the imaging data. For example, the imaging data could be color-coded to reflect the classification of the sensor data.

1.2 Problems in Sensor-Integrated Robotics

Multiple capabilities are needed to have an effective sensor-integrated robotic system. These capabilities are in the areas of sensors, robotics, and techniques for robot/sensor integration. For example, the capabilities may involve processing sensor signals or accurately positioning a sensor. For many types of applications, there are shortcomings in the current technologies employed to provide the capabilities. These issues make it harder to take advantage of sensor-integrated robots and possibly constrain their adoption. In the following sections, some problem areas are discussed in which improvements would be highly valuable.

1.2.1 Sensor Signal Analysis Problems

Sensor technology has become more sophisticated over the years. Sensors are now used to measure quantities or identify differences that occur at a very small scale. For instance, one may wish to examine trace amounts of dangerous chemicals or find disparities between normal and abnormal skin cells. When the properties of interest are so subtle, it often becomes difficult to obtain clear data from a sensor. There may be noise and other undesirable quantities mixed into the data. For example, a Raman spectrum typically has background fluorescence, shot noise, cosmic ray spikes, and more.

To help overcome this problem, computer algorithms based on mathematical/statistical techniques are typically employed to reduce noise/distortions and find meaningful patterns in the collected data. There is even an entire field (called chemometrics) devoted to the use of such methods to extract information from chemical data. The algorithms for processing and analyzing sensor data are generally very complex, and they are not guaranteed to work under all circumstances. Thus, there is always a need for improved algorithms. In addition, researchers and organizations often develop custom software systems for their algorithms, leading to standard procedures being repeatedly implemented and novel techniques being harder to share. The development of more open and accessible software platforms would help prevent these issues.

1.2.2 Integrated Robot/Sensor Problems

Simply placing a sensor on the end-effector of a robot may not be sufficient for many applications. There are a number of issues that remain, including some that relate to the user interface. One issue is that a system may be required to help position the robot arm and attached sensor in order to collect the desired data. For example, a surgical robot equipped with a Raman

sensor would need to be positioned very carefully in order to collect data from the tissue of interest without disturbing or damaging structures inside the patient. Another issue is that it may be difficult for the user to visualize the relationship between collected sensor data and the environment from which it was collected.

Image guidance can be used to help solve these problems. In the past, image-guided systems have been used to track manipulators within an environment. However, these systems have not typically incorporated analytical sensors (such as a Raman probe) and the information they provide. The development of image guidance systems that support advanced analytical sensors could greatly enhance the usage of sensor-integrated robotic systems.

1.2.3 Robot Accuracy Problems

Although robotic systems have the capability to position a manipulator or sensor with great precision, the accuracy depends upon the quality of the model used to control the robot (among other things). This is because a robot is typically moved using a model of its kinematic properties. For example, a model has to include the type of each joint, the lengths of the links connecting the joints, the changes in orientation between joints, etc. In addition, any tools or sensors being handled by the robot must be incorporated into the model. Any differences between the model and the actual robot lead to errors when positioning the robot.

Even small differences can cause large problems because the effects of an error can be magnified as calculations progress from a robot's base to its end-effector. As a simple example, imagine a 1 m link with one end that rotates about a fixed point. If the rotation is off by just 0.01 rad, the position of the other end of the link will be off by approximately 1 cm.

Due to errors/tolerances in a robot's construction or possibly the wear of usage, a robot will vary from its manufacturer's specifications. Thus, a technique is needed to correct or

compensate for differences between the actual robot and its model. For a robot's kinematic properties, this process is known as kinematic calibration. A properly executed kinematic calibration can greatly increase the accuracy of a robot if there are significant errors in its model.

There are many techniques and algorithms for kinematic calibration. Some techniques require external measurement devices (often expensive or complex), but some need only the internal joint feedback of the robot. Although calibration is simpler without an external measurement device, several issues arise due to limitations of the collected data. For example, it may not be possible to calibrate certain parameters, the model might have the wrong scale, and the position/orientation of the robot's base may not be ascertainable. The development of flexible calibration techniques that help overcome these problems (while retaining ease of execution) would be useful for improving the accuracy of robots.

1.3 Aim and Scope of Research

The overall aim of this research was to develop novel techniques and systems that facilitate the use of sensor-integrated robotic systems. To accomplish this aim, solutions to the three sets of problems described under section 1.2 were developed. Because the problems are somewhat abstract and can appear in many different kinds of applications, some specific technologies/systems were chosen for the developed solutions.

Raman spectroscopy was selected for the sensor-related work in this dissertation, with a focus on chemometric analysis (extracting information from chemical data). There are several reasons for this choice. First, Raman spectroscopy is very powerful sensing technique with numerous applications, such as chemical composition analysis [11, 12] and cancer identification (both *ex vivo* and *in vivo*) [13-26]. Second, complex algorithms are required to process Raman spectra. Moreover, there is a lack of an open, easy-to-use platform for analyzing Raman spectra.

Third, a Raman probe requires steady and accurate positioning, which makes it especially suitable for robotic manipulation and image guidance.

An image-guided surgery system was chosen to serve as the prototype for a sensor-integrated robotic system. The surgery system could allow either manual or robotic positioning of the sensor. One reason for selecting this kind of system is that navigating the environment of a patient's body is a difficult task that is well suited to the use of image guidance. In addition, integrated sensors (such as a Raman probe) could provide an abundance of valuable information about tissues of interest.

With respect to techniques to improve robot accuracy, a decision was made to work on serial manipulators. This type of robot is very appropriate for positioning a sensor in an environment. Furthermore, a choice was made to focus on kinematic parameters (rather than dynamic/inertial parameters) and position accuracy (with less emphasis on orientation accuracy). These things are very important concerns when it comes to accurately positioning a sensor, and it is easier to obtain measurements for them. Lastly, it was decided to work with robot models expressed in the Denavit–Hartenberg convention [27, 28] because it is a very common way to represent a robot's structure.

1.4 Specific Research Objectives

As discussed above, there were three sets of problems in sensor-integrated robotics that this work sought to overcome. Thus, the research was divided into three specific objectives, one for each of the problem areas. Figure 1.1 provides an overview of these objectives, and they are described in more detail below.

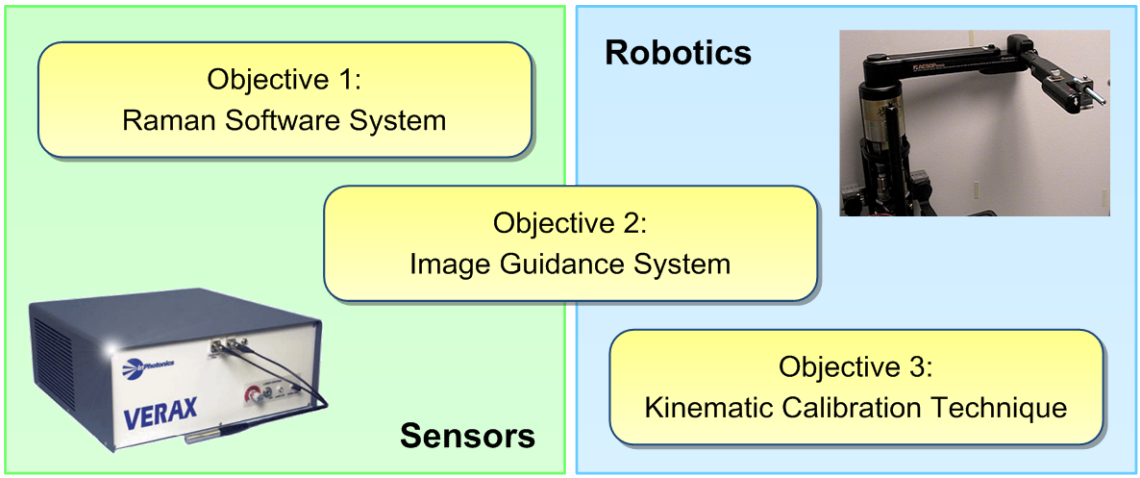

Figure 1.1: Specific research objectives and the technologies to which they apply.

The **first research objective** was to develop a software system that processes, analyzes, and classifies data from a specific kind of sensor (a Raman spectrometer). The system had to be open, easily accessible, and extensible, which would enable interested parties to utilize and contribute to a common implementation of relevant algorithms. In addition, it was desired to implement some novel techniques for processing and analyzing the data collected by the sensor.

The **second research objective** was to develop an image guidance system for use with a sensor-integrated robotic system (a Raman probe attached to a surgical system). The image guidance system needed to support tracking and displaying the sensor's location using computer visualization with imaging data of the environment. Furthermore, the system had to enable the acquisition and real-time analysis of sensor data (using techniques from the first objective), and it needed the ability to present the results within the visualized representation of the environment.

The **third research objective** was to develop a kinematic calibration technique for serial manipulators. The method had to be able to enhance the positional accuracy of a robot, which is important for sensor placement. It was desired that the technique be relatively simple to apply and not require any external measurement devices for the calibration. In addition, the work was

expected to identify and provide solutions for some limitations of existing techniques that utilize similar information.

1.5 Novelty and Significance

The purpose of this work was to advance the state of robotic systems and integrated sensors. To achieve this aim, a number of novel techniques and systems were developed. These contributions were made in three different areas, corresponding to the three specific objectives of the research.

1.5.1 Sensor Signal Analysis

With respect to sensor signal analysis, a unique software system for processing, analyzing, and classifying Raman spectra was developed. The system possesses a simple graphical interface that enables anyone to manage groups of spectra and apply powerful algorithms. These algorithms include novel techniques for subtracting artifacts from spectra and automatically identifying important peaks/components in spectra. Moreover, some of the algorithms are suitable for real-time classification of newly acquired spectra.

The software was designed as a platform that can be easily extended. In addition, its source code has been made freely available to the research community. These properties will allow others to add new features and algorithms to the system and avoid redundant implementations of existing techniques. In fact, the system is already being used by other researchers for the chemometric analysis of Raman spectra.

The software system has numerous potential applications in the analysis and classification of various substances. For example, the system is already being used to distinguish cancerous tissue from healthy tissue. Cancer is a significant health issue, with an estimated 12.7 million

new cases and 7.6 million deaths occurring in 2008 [29]. By using the system to aid surgeons in the accurate removal of tumors, it is hoped that lives may be prolonged or saved.

1.5.2 Robot/Sensor Integration

Regarding robot/sensor integration, a system that provides image guidance for a Raman probe was created. To the author's knowledge, it was the first with this capability. The system supports real-time tracking of the probe (or other tools) using virtual or augmented reality visualization that incorporates imaging data of the environment. This enables accurate placement of the sensor using either a manual or a robotic positioning system.

In addition, the system enables the near-real-time collection and analysis of Raman spectra as well as the presentation of the results. A method was produced to integrate the results with the imaging data in the computer visualizations to aid the user in understanding the environment or specimens of interest. The system is flexible enough to support different tracking/positioning devices, and it can produce visualizations on multiple displays.

The system was conceived as a prototype for eventual use in image-guided surgery. With further development and testing, it is hoped that the system will enable the *in vivo* use of a Raman probe. The integrated software and hardware solution should empower a surgeon to quickly identify diseased tissue (e.g. cancer) and immediately provide the appropriate treatment. Such a system could save lives and ultimately reduce costs.

1.5.3 Robot Accuracy

Concerning robot accuracy, a technique for the kinematic calibration of serial manipulators was developed. The technique is flexible and easy to execute. Although the idea of using internal joint measurements from a fixed point is not new, there are some limitations in existing techniques. These issues were identified, and methods that solved or mitigated them were

produced. For instance, a registration process is used to overcome limitations that arise due to the lack of absolute end-effector measurements.

The calibration technique was developed to help fulfill a need for accuracy in the robotic placement of a sensor. The technique should make it easier to reduce the positional error of a robot, thus improving the quality of data obtained from a sensor. For example, a tumor could be located with greater accuracy through improved sensor placement, and the treatment could be applied with better precision. The technique can also be executed on robots without analytical sensors or on arms without active joints, which means it could provide benefits in numerous and diverse applications.

1.5.4 Combined Impact

Together, the contributions of this work provide important building blocks for an accurate robot with an integrated Raman probe. This type of a system would be a powerful tool for many future applications. For example, a surgical robot could automatically scan for diseased tissue and assist the surgeon in the necessary treatment. A rover could provide fast sampling and a detailed analysis of extraterrestrial soil samples, or a military robot could be programmed to search a hostile environment or vehicle for explosives. Although the work was developed using a Raman probe, much of it is generally applicable to different analytical sensors. Thus, the work can serve as a model for the design of advanced sensor-equipped robotic systems in many other applications.

1.6 Organization of the Dissertation

The remainder of this dissertation is organized in the following manner. Chapter 2 provides general background information on some topics that are found throughout the dissertation. In Chapter 3, the developed Raman software system and its capabilities are described. Chapter 4

presents the image guidance system that was produced for a Raman probe. Next, Chapter 5 details the developed kinematic calibration technique and the results of several experiments. Finally, Chapter 6 offers a discussion of the work and potential directions for future research.

Chapter 2:
General Background

2.1 Overview

This chapter provides some general background material that serves as a foundation for other portions of the dissertation. Specifically, the information on Raman spectroscopy (section 2.2) is relevant to Chapter 3 and Chapter 4, and the material on robot modeling (section 2.3) is used in Chapter 4 and Chapter 5.

2.2 Raman Spectroscopy

Raman spectroscopy is an optical sensing technique that relies on the inelastic scattering of light by a substance [30]. It provides information about the molecular or chemical composition of a specimen. Its measurements are fast, simple, accurate, and nondestructive, making it a popular choice for the analysis of myriad substances.

In Raman spectroscopy, a beam of intense monochromatic light (typically a laser) is shined upon a substance. The light is then scattered by the substance in one of two ways. Most of the photons of light have an elastic collision, resulting in scattered light that has the same energy as the incident light. This is known as Rayleigh scattering. However, approximately one photon in a million will exchange energy with the substance, resulting in scattered light with a different energy (frequency) than the incident photons. This is called Raman scattering [30, 31]. If a photon loses energy to the substance, it is known as Stokes Raman scattering. If the photon gains energy, which happens much less frequently, it is called anti-Stokes Raman scattering.

The difference in energy between an incident and scattered photon is equal to the change in the vibrational energy of the struck molecule. This vibrational energy depends on the bonds and

other properties of the molecule. Thus, the shift in energy (and thus frequency) of a photon corresponds to the type of molecule being struck. By measuring the energies of all the photons scattered from a substance, a great deal of information about the molecular (or chemical) makeup of the substance is obtained.

A Raman spectrometer measures the Raman-scattered light over a range of frequencies to produce a Raman spectrum (Figure 2.1). The *x*-axis indicates the shift in wavenumber from the monochromatic light source. (A wavenumber is the inverse of the wavelength, which is proportional to the frequency and energy of the Raman shift.) The *y*-axis shows the intensity of scattered light at each wavenumber. A typical Raman spectrum consists of bands of intensity around various wavenumbers, which indicate the molecular makeup of the substance. Thus, a Raman spectrum acts like a fingerprint for a specimen.

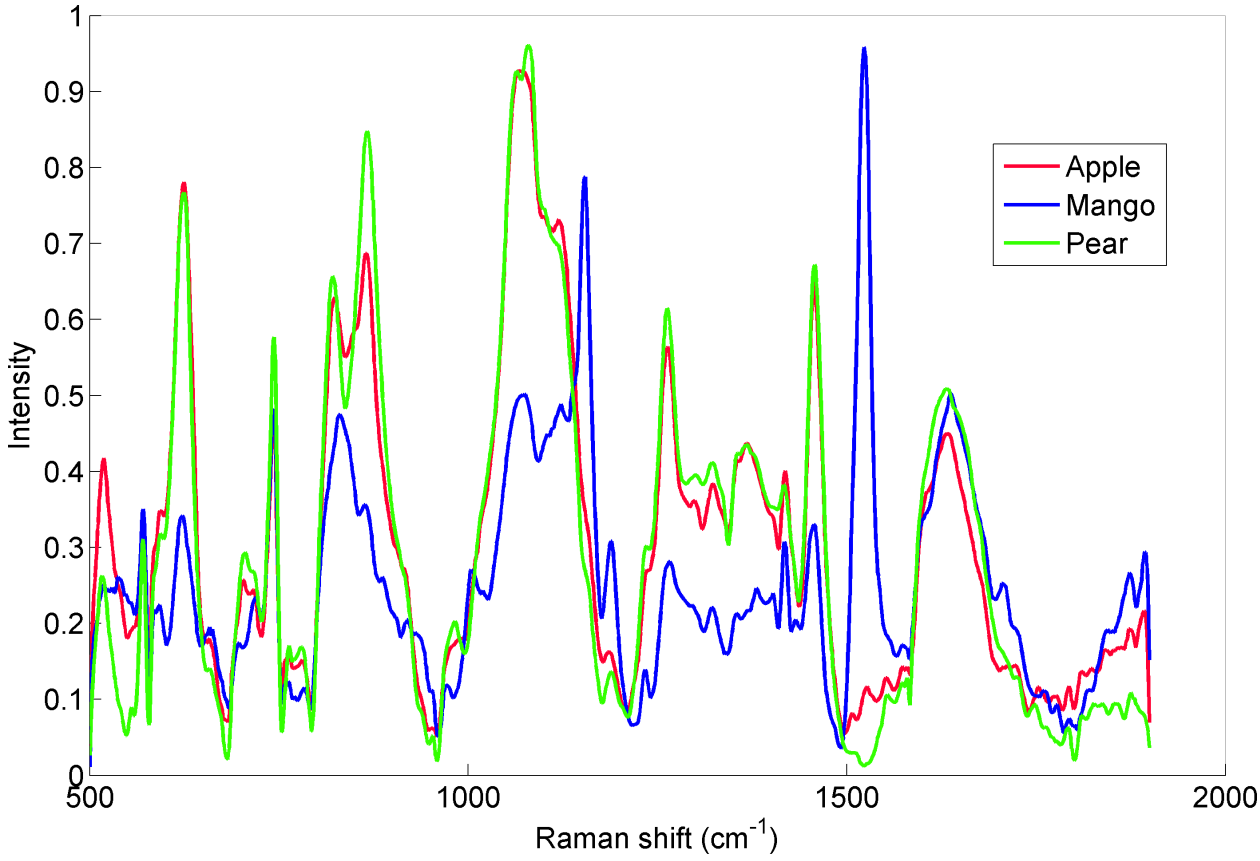

Figure 2.1: Raman spectra from three different fruits.

2.2.1 Raman Applications

Raman spectroscopy is used in numerous applications, such as the analysis of nanomaterials [32, 33], viruses/microorganisms [34, 35], and industrial processes [36, 37]. It is frequently used to determine the composition of a mixture of chemicals [11, 12]. A Raman system can also be used to detect explosives at a distance [7]. In addition, Raman spectroscopy has started to be used for medical analysis [38, 39]. In particular, it holds great promise for cancer detection [40]. It has already been shown to be able to identify cancers of the skin, breast, and other tissues [22-26].

Raman spectroscopy is particularly suited for *in vivo* cancer diagnosis [41]. It is quick, precise, and non-destructive, and it requires no sample preparation. Furthermore, fiber optic

probes enable measurements to be taken in a minimally invasive fashion. Several research groups have been working towards *in vivo* Raman spectroscopy, including the Massachusetts Institute of Technology [13, 14], the Erasmus Medical Center [15, 16], the University of Texas at Austin [17, 18], and Wayne State University [19-21].

2.2.2 Analysis of Spectra

As mentioned above, a Raman spectrum is a plot of the Raman-scattered light intensity against the wavenumber shift. The intensity at each point along the range of wavenumbers can be representative of a molecular component of the specimen. The wavenumber range often spans the "fingerprint" region, 500–1800 cm^{-1}, because it contains most of the distinctive Raman scatter from organic molecules [42]. A typical Raman spectrometer has a resolution of a few wavenumbers, so it is common for a captured Raman spectrum to contain hundreds of data points. Within those data points, a significant amount of components may be represented, some of which may overlap.

The large number of data points and components in a Raman spectrum makes it difficult for a human or computer to analyze. In addition, the analysis of a spectrum is complicated by the presence of noise, background fluorescence, non-ideal instrument responses, different environmental conditions, etc. As a result, it can be difficult to determine exactly what each part of Raman spectrum represents. For the same reasons, it is hard to establish the differences (and their causes) between two Raman spectra captured under distinct conditions.

To aid in the analysis of such complicated data, the field of chemometrics was established [43]. Chemometrics involves the application of mathematics and statistics to the investigation of chemical information. It actually includes a variety of techniques, such as signal processing,

calibration, factor analysis, and pattern recognition. A few of the techniques relevant to the study of Raman signals will be presented in the following section.

Chemometrics is already being applied to assist in the analysis and classification of Raman spectra. Some of the most common techniques used in the literature are principal component analysis [25, 26, 44] and least squares analysis [14, 45]. These methods can be employed to determine important parts of a spectrum, represent a spectrum in a more compact form, or to model the components present in a spectrum.

Modeling the components in a spectrum is one particularly useful subset of chemometrics. A calibration model is used to establish the relationship between a substance's components and its Raman spectra. There are a large number of papers that discuss methods to apply when a calibration model needs to be transferred to different equipment [46, 47]. These methods often involve changing the calibration model or the Raman spectra to account for differences in the wavelengths or intensities. More general changes in Raman spectra, such as those caused by different environments or transformations in the specimen itself, could benefit from similar methods. With significant preparation, some calibration models have been made for specific tissues [39, 48].

Changes in Raman spectra can be expected to occur in three different circumstances [46]. The situations are when the Raman equipment changes, when the environment changes, or when the specimen itself changes. For example, spectral changes will occur as human tissue transforms from normal to cancerous. All of these situations can result in additive intensity changes, multiplicative intensity changes, or wavelength shifts in the Raman spectra. Careful analysis is needed to identify these changes because they are often obscured by noise or background fluorescence, and they may occur simultaneously.

2.2.2.1 Common Techniques

Principal component analysis (PCA) is a statistical technique used to reduce the dimensionality (size) of complex data sets [49]. It is applied when the data variables have complex relationships to each other. PCA transforms the data into new, uncorrelated variables (principal components) based on the variation in the data set. Only a few principal components are needed to capture most of the variation. Thus, PCA provides a compact representation of data that can be used as an input for other (e.g., classification) algorithms.

Regression analysis is used to establish relationships between two or more variables [50]. The goal is to determine how a dependent (response) variable varies as a function of one or more independent (predictor) variables. For example, the response variable could be the concentration of a certain chemical, and the predictor variables could be a Raman spectrum. If the response variable is a linear function of the predictor variables, it is called linear regression. In simple linear regression, only one predictor variable is considered, whereas as in multiple linear regression (MLR), multiple predictor variables are taken into account.

Principal component regression (PCR) is a form of MLR [49]. In ordinary MLR, correlated variables make it difficult to establish the dependent/independent variable relationships. PCR uses PCA to remove the correlation (and maximize the covariance) in the predictor variables so that the regression can proceed more smoothly.

Another common form of MLR is partial least squares regression, or PLSR [51]. It can be viewed as an improvement upon PCR. Instead of simply maximizing the covariance of the predictor variables, PLSR tries to maximize the covariance between the response and predictor variables. PLSR is more complicated than PCR, but it can produce a better model of the relationship between the response and predictor variables.

One popular algorithm used to transfer a calibration model to different equipment is called direct standardization (DS) [52]. It can transform a spectrum taken on one Raman spectrometer so it looks like it came from another. It can correct for both multiplicative intensity changes and wavelength shifts. An extended version of DS, called piecewise direct standardization (PDS), corrects the spectrum in smaller chunks of wavelengths to reduce the computational burden. Various improvements have been made to DS and PDS, such as a version that attempts to eliminate background differences (additive intensity changes) in the spectrum [53].

2.2.3 Classification of Spectra

Classification algorithms are used to assign data (such as a Raman spectrum) to a particular class out of a set of two or more classes. For instance, a classification algorithm can be used to distinguish the Raman spectra of healthy and cancerous tissues. As another example, one could be used to associate a bacteria sample with one of multiple species or strains. A classification algorithm is a key part of any system that is intended to identify a specimen.

Classification algorithms fall under the umbrella of machine learning [54]. They use statistical classification techniques to place items into groups based on known information. The process of teaching a classifier to recognize items is called learning or training. Classifying new items based on the learned information is known as recall.

There are two different types of learning. In unsupervised learned, the items have no predefined membership in any classes. Thus, the items are clustered into groups based on similarities. In supervised learning, the classes of items are known ahead of time. Therefore, the system is trained to recognize items as belonging to their pre-assigned classes.

The data used to teach a supervised classification algorithm is known as a training set. The training set consists of a large number of items (e.g., Raman spectra) from each class. The items

assigned to each class should be representative of future items that one would like to assign to that class. If they are not (perhaps due to incorrect initial classification), the classifier's performance will be reduced.

There are many different types of classification techniques. Some come directly from the field of statistics, whereas others are from computer science. They use a wide variety of techniques to learn from existing data and classify new data. For example, some methods are deterministic, while others rely on randomization and iterative optimization. Several commonly used classification methods will be described in the following section.

Classification algorithms, particularly supervised ones, are often used to categorize Raman spectra. Linear discriminant analysis is one of the most popular techniques, especially for cancer detection [55, 56]. Artificial neural networks and nearest neighbor techniques are also commonly used [20, 57]. More recently, support vector machines have been applied for Raman classification as well [57, 58].

2.2.3.1 Common Techniques

Nearest neighbor classifiers represent one of the simplest classification techniques. They work by comparing the distance between a new piece of data and already classified data. The distance measure is often a Euclidean distance in a feature space derived from the data (e.g., through PCA). In the *k*-nearest neighbor algorithm, the new data is assigned to the most represented class among the classes of its closest *k* neighbors [54]. This technique doesn't really require training, and most computation is done when classifying a new item. Due to the number of comparisons required, this technique is typically used with relatively small data sets.

Linear discriminant analysis (LDA) is a statistical technique that is used to find the variables that best discriminate between two or more groups [59]. It attempts to find linear combinations

of the input variables that maximize the distances between groups while minimizing the differences within groups. After calculating these combinations, the information can be used to produce a different classification function for each group. Running an input through each function produces a classification score; the input is classified as belonging to the group corresponding to the highest score.

Artificial neural networks (ANNs) are parallel computational models based on biological neural networks [60]. They consist of interconnected processing units called neurons that produce outputs based on a weighted function of their inputs. There are multiple types of ANNs, including perceptrons, radial basis function networks, and recurrent networks. Training is often accomplished through an iterative technique, such as a gradient descent of the network parameters. ANNs can overcome a certain amount of noise and distortion, but care must be taken to ensure that the network is generalized enough to work on inputs not contained in the training set. Once trained, a neural network is a relatively fast classifier.

Support vector machines (SVMs) are a set of techniques for pattern identification that have recently become more popular [61]. They use hyperplanes to separate data in a high dimensional space. SVMs were originally used as linear classifiers, but they are often adapted (using kernel functions) for nonlinear separation. They are also inherently binary (two-class) classifiers, but multiple SVMs can be employed to support multiple classes. A trained SVM has its separating hyperplane positioned to achieve the largest possible distance from both of the training classes. This makes it easier for an SVM to attain the generalization that is needed to perform well on new inputs.

2.3 Robot Modeling

A serial robotic manipulator typically consists of a sequence of points of interest (the base of the robot, its joints, and the tip of the end-effector) that are connected by links. To accurately position a robot, a mathematical model that represents this physical structure is needed. The model must capture certain information, such as the position of each point of interest, the orientation of the object at each point of interest, the type of each joint (e.g., revolute or prismatic), and the value of each joint (e.g., its angle of rotation).

A variety of conventions exists to model a robot's physical structure. For this work, two forms of the commonly used Denavit–Hartenberg (DH) convention were chosen [27, 28]. There is the original form introduced by Denavit and Hartenberg [27], and there is a modified form described by Craig [28]. Although the two forms are conceptually very similar, they are not directly compatible. Therefore, this dissertation uses the terms "original" and "modified" to distinguish between the two forms.

The following subsections describe some important concepts using the DH convention. First, a more detailed explanation is given for the original form of the DH convention. Then, a shorter summary is provided to highlight some important changes when using the modified form of the DH convention.

2.3.1 The Original Denavit–Hartenberg Convention

The DH convention provides a compact way to represent the structure of a robot. Each point of interest is assigned a coordinate frame that defines the position and orientation (pose) of the object (e.g., joint/link or end-effector) at the point of interest. This coordinate frame is fixed to the object at the point of interest, even during movements of the robot. Therefore, as the object moves, so does the coordinate frame. The DH convention uses a set of four parameters per

coordinate frame, called DH parameters, to describe the position and orientation of each coordinate frame relative to the previous coordinate frame. Thus, the sets of DH parameters are the convention's mechanism for expressing the structure of a robot.

The four types of DH parameters are the link twist, link length, joint angle, and link offset. There is one set of these parameters for each coordinate frame (other than the base frame). In the original DH convention, the parameters have the following definitions:

- Link twist (α_i) represents the angle from the z_{i-1} axis to the z_i axis about the x_i axis
- Link length (a_i) indicates the distance between the z_{i-1} and z_i axes along the x_i axis
- Joint angle (θ_i) specifies the angle from the x_{i-1} axis to the x_i axis about the z_{i-1} axis
- Link offset (d_i) denotes the distance from the x_{i-1} axis to the x_i axis along the z_{i-1} axis

Note that in the original DH convention, the origin of coordinate frame i is actually located at joint i+1. Figure 2.2 shows a model of the AESOP 1000 robot (Intuitive Surgical, Sunnyvale, CA) expressed using original DH parameters.

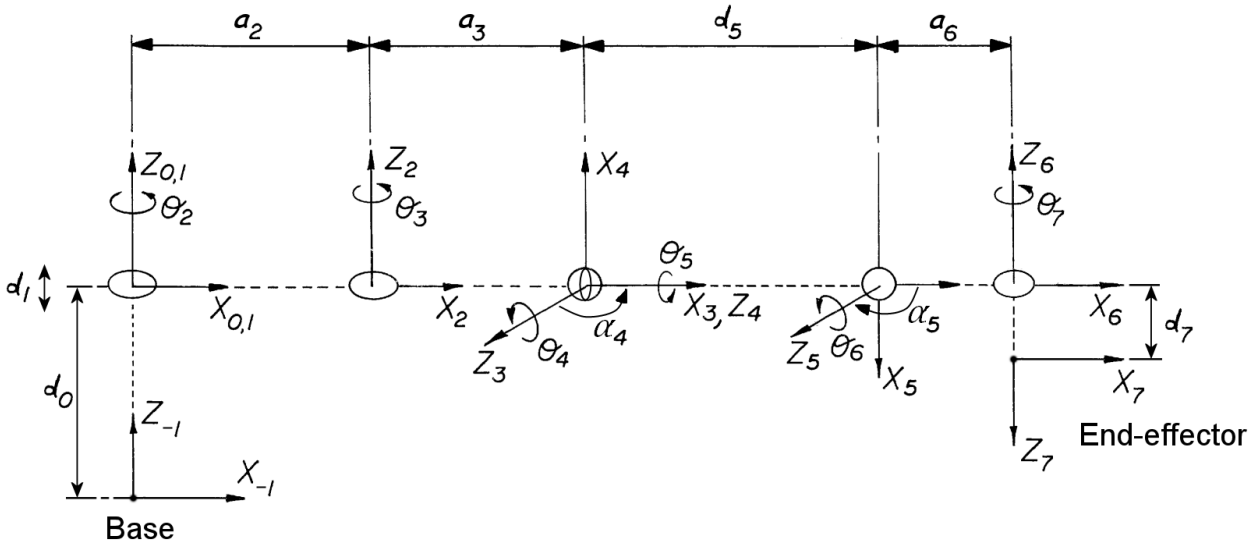

Figure 2.2: Model of the AESOP 1000 robot expressed in the original DH convention.

For revolute joints, the joint angle (θ_i) parameter is a variable that represents the joint's angle of rotation. For prismatic joints, the link offset (d_i) parameter is a variable that indicates the joint's amount of extension. For either type of joint, the variable parameter can be split into a variable portion (θ_i^{var} or d_i^{var}) and a fixed offset (θ_i^{off} or d_i^{off}) according to Equation 2.1 or Equation 2.2. The fixed offset can be used to correct any discrepancy between the joint's assumed zero position and the actual zero position. Although the offset does not directly reflect the geometric structure of the robot, it does alter the meaning of any measured joint values.

$$\theta_i = \theta_i^{\text{var}} + \theta_i^{\text{off}} \quad \text{Equation 2.1}$$

$$d_i = d_i^{\text{var}} + d_i^{\text{off}} \quad \text{Equation 2.2}$$

2.3.1.1 Transformations

It is necessary to introduce a way to express a relative transformation from one coordinate frame to another. In particular, the representation must be able to express both the change in orientation and the change in position. For this purpose, a 4×4 homogeneous transformation matrix can be used [28, 62]. The following equations define the structure of a homogeneous transformation matrix and two of its subcomponents.

$${}^{\text{A}}\boldsymbol{T}_{\text{B}} = \begin{bmatrix} r_{x,1} & r_{y,1} & r_{z,1} & p_x \\ r_{x,2} & r_{y,2} & r_{z,2} & p_y \\ r_{x,3} & r_{y,3} & r_{z,3} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{Equation 2.3}$$

$${}^{\text{A}}\boldsymbol{R}_{\text{B}} = \begin{bmatrix} r_{x,1} & r_{y,1} & r_{z,1} \\ r_{x,2} & r_{y,2} & r_{z,2} \\ r_{x,3} & r_{y,3} & r_{z,3} \end{bmatrix} \quad \text{Equation 2.4}$$

$${}^{\text{A}}\boldsymbol{p}_{\text{B}} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \quad \text{Equation 2.5}$$

In Equation 2.3, ${}^{\mathrm{A}}\boldsymbol{T}_{\mathrm{B}}$ represents the relative transformation from some coordinate frame, A, to another coordinate frame, B. The upper-left 3×3 submatrix of ${}^{\mathrm{A}}\boldsymbol{T}_{\mathrm{B}}$ is a rotation matrix, ${}^{\mathrm{A}}\boldsymbol{R}_{\mathrm{B}}$, that defines the orientation of frame B with respect to frame A. Specifically, the column vectors of ${}^{\mathrm{A}}\boldsymbol{R}_{\mathrm{B}}$ are unit vectors that represent the directions of the coordinate axes of frame B when expressed in the coordinate system of frame A. The upper-right 3×1 submatrix of ${}^{\mathrm{A}}\boldsymbol{T}_{\mathrm{B}}$ is a vector, ${}^{\mathrm{A}}\boldsymbol{p}_{\mathrm{B}}$, that provides the position of the origin of frame B in terms of the coordinate system of frame A. In other words, the origin of frame B is located at coordinates (p_x, p_y, p_z) in the coordinate system of frame A.

A 3×3 matrix representation of a rotation consists of nine values, which is more than necessary. Sometimes, a more compact form is preferred. Any change in orientation can be minimally represented by a set of three Euler angles (φ, θ, ψ). There are multiple conventions for Euler angles, so this dissertation will use the *z*-*y*′-*z*″ convention [63].

In the *z*-*y*′-*z*″ convention, three steps are required to derive the change in orientation of a coordinate frame from the Euler angles. First, the coordinate frame is rotated by φ about the *z*-axis. Next, the frame is rotated by θ about its new *y*-axis (*y*′). Finally, the frame is rotated by ψ about its new *z*-axis (*z*″). To convert the Euler angles to an equivalent 3×3 rotation matrix, Equation 2.6 can be used. The conversion from a rotation matrix to Euler angles can be found in [63].

$$\boldsymbol{R}_{\varphi,\theta,\psi} = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{Equation 2.6}$$

Equation 2.3 has a fourth row because the transformation matrix operates using homogeneous coordinates. This representation, which adds an additional coordinate, allows transformation matrices to be easily combined. Suppose that the matrices are known for a

transformation from frame A to frame B ($^{A}\boldsymbol{T}_{B}$) and a transformation from frame B to frame C ($^{B}\boldsymbol{T}_{C}$). To obtain the matrix for the transformation from frame A to frame C ($^{A}\boldsymbol{T}_{C}$), one can simply multiply the transformation matrices as shown in Equation 2.7. In other words, this equation produces a transformation matrix that expresses frame C in the coordinate system of frame A. As explained in the following section, this property provides an elegant solution for the calculation of forward kinematics.

$$^{A}\boldsymbol{T}_{C} = {}^{A}\boldsymbol{T}_{B}\,{}^{B}\boldsymbol{T}_{C} \qquad \text{Equation 2.7}$$

2.3.1.2 Forward Kinematics

The set of four DH parameters for a specific coordinate frame only provides the position and orientation of the frame with respect to the previous frame. Thus, the DH parameters correspond to a relative transformation. However, it is often desired to know the position and orientation of any coordinate frame with respect to the base frame. The procedure by which the position and orientation of any coordinate frame (especially the end-effector frame) is expressed in the base coordinate system is known as forward kinematics.

A homogeneous transformation matrix can be formed for each set of four DH parameters. This matrix describes the change in position and orientation produced by the four parameters, including the effect of the variable joint value. For parameters expressed in the original form of the DH convention, Equation 2.8 shows how to generate the transformation matrix [27].

$$^{i-1}\boldsymbol{T}_{i} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Equation 2.8}$$

Here, ${}^{i-1}\boldsymbol{T}_i$ indicates the transformation from coordinate frame $i-1$ to frame i. The change in orientation is determined by the values of the angular parameters, θ_i and α_i. The change in position is a function of the length parameters, a_i and d_i, as well as θ_i.

Homogeneous transformation matrices can be generated for every set of four DH parameters that describe a given robot. These computations will produce $\{{}^{0}\boldsymbol{T}_1, {}^{1}\boldsymbol{T}_2, \dots, {}^{n-1}\boldsymbol{T}_n\}$, assuming frame 0 is the base frame and frame n is the last (end-effector) frame. Note that each of these transformations is relative to its previous transformation. To express any frame in terms of the base coordinate system rather than the coordinate system of the previous frame, an additional calculation must be made.

Equation 2.7 shows how to combine two adjacent, relative transformations, but it can be applied multiple times to combine multiple adjacent transformations. By multiplying a series of relative transformation matrices, it is possible to determine a single transformation from the first frame to the last frame. When applied to the transformation matrices of a robot, this enables the calculation of the pose of any coordinate frame in terms of the base coordinate system. For example, Equation 2.9 shows how to calculate the position and orientation of the end-effector (frame n) in terms of the base coordinate system (frame 0).

$$ {}^{0}\boldsymbol{T}_n = {}^{0}\boldsymbol{T}_1 {}^{1}\boldsymbol{T}_2 \cdots {}^{n-1}\boldsymbol{T}_n = \prod_{k=1}^{n} {}^{k-1}\boldsymbol{T}_k \qquad \text{Equation 2.9} $$

The position (${}^{0}\boldsymbol{p}_n$) and orientation (${}^{0}\boldsymbol{R}_n$) of the end-effector in the base coordinate system can be readily extracted from the computed transformation, ${}^{0}\boldsymbol{T}_n$. In addition, the pose of any other frame can be found in terms of the base coordinate system by ending the multiplications when the desired frame is reached. Thus, Equation 2.9 represents the computation of forward kinematics.

2.3.2 Craig's Modified Denavit–Hartenberg Convention

An alternative form of the DH convention was introduced by Craig in [28]. It is almost the same as the original convention, but it differs slightly in the frame assignments for the parameters. Specifically, the origin of coordinate frame i is moved to the location of joint i (instead of joint i+1). This results in the DH parameters having the following modified definitions:

- Link twist (α_i) represents the angle from the z_i axis to the z_{i+1} axis about the x_i axis
- Link length (a_i) indicates the distance between the z_i and z_{i+1} axes along the x_i axis
- Joint angle (θ_i) specifies the angle from the x_{i-1} axis to the x_i axis about the z_i axis
- Link offset (d_i) denotes the distance from the x_{i-1} axis to the x_i axis along the z_i axis

For both the original and modified conventions, the joint angle (θ_i) and link offset (d_i) are measured from frame $i-1$ to frame i. However, the link twist (α_i) and link length (a_i) in the modified convention are measured from frame i to frame i+1. In order to group all the parameters with the same frames, DH parameters for the modified convention are usually reported in the following fashion: $\{\alpha_{i-1}, a_{i-1}, \theta_i, d_i\}$. This manner of grouping is used for the modified DH parameter tables in this dissertation.

Due to the change in parameter definitions, some equations used to calculate the kinematics of a robot must also be changed. For this research, the changes with respect to forward kinematics are of concern. To calculate the forward kinematics of a robot model expressed in Craig's modified DH convention, Equation 2.8 is no longer valid. Equation 2.10 provides the new formula to calculate the homogeneous transformation matrices that represent the changes in position/orientation between adjacent coordinate frames [28]. Note that Equation 2.9 is still valid for the final computation of the forward kinematics.

$$ {}^{i-1}\boldsymbol{T}_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i \cos\alpha_{i-1} & \cos\theta_i \cos\alpha_{i-1} & -\sin\alpha_{i-1} & -d_i \sin\alpha_{i-1} \\ \sin\theta_i \sin\alpha_{i-1} & \cos\theta_i \sin\alpha_{i-1} & \cos\alpha_{i-1} & d_i \cos\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$ Equation 2.10

Chapter 3:
Raman Software System

3.1 Introduction

To effectively utilize the data collected by an analytical sensor, significant signal processing is generally required. This type of processing is typically performed using software that runs on a computer system. The software helps to extract meaningful information from the data and use it to make conclusions about the sample being examined.

As explained in Chapter 1, a Raman spectrometer was chosen to demonstrate the sensor-related concepts of this dissertation. Therefore, this chapter describes a sensor data processing system that is intended to work on Raman spectral data. The system is an updated version of the one presented in [64].

Although the system was customized for Raman applications, it uses many techniques that are commonly applied to data from other types of analytical sensors. For example, noise reduction algorithms are used to improve the quality of many kinds of signals, and classification algorithms can be applied to help categorize almost any data. Thus, with sufficient work, the system presented in this chapter could be adapted for other types of analytical sensors.

3.2 Overview

Sensors can be powerful tools for analyzing various substances. However, a significant amount of processing is often required to extract useful information from the raw data that is collected. This is certainly the case for spectra that are collected via Raman spectroscopy. Many research groups develop custom software systems to process spectra, which leads to standard algorithms being repeatedly implemented and novel techniques being difficult to share. This

chapter presents an integrated software system that processes, analyzes, and classifies Raman spectra. The system is freely available (http://cares.wayne.edu/rp/), open, cross-platform, and extensible, enabling interested parties to utilize and contribute to a common implementation of relevant algorithms.

A wide variety of useful features are included, such as noise filtering, automated background fluorescence subtraction, principal component analysis, linear discriminant analysis, artificial neural network classification, and support vector machine classification. The system supports customizable groupings of related spectra and is able to retrieve spectra using a searchable database. Everything is executed through a simple graphical user interface.

The developed system was used to process two data sets containing different types of spectra. The results demonstrate that the system is capable of fast, accurate processing of Raman spectra. The open system will become even more powerful if other developers are willing to add new and improved features. In addition, it is intended that the freely available system will make Raman spectroscopy more accessible to all researchers and promote progress in the field.

3.3 Acknowledgements

The following individuals contributed to the implementation of the Raman software system: Samir Al-Stouhi, Alex Cao, Alan Fung, Rachel Kast, Bradley Martens, and Abhilash Pandya. Note that their contributions to the portions of the system described in this chapter were limited to implementations of already published techniques. All the novel techniques that are discussed (2D morphological background subtraction, artifact removal, significant peak characterization, etc.) were implemented by the author.

3.4 Background and Motivation

Raman spectroscopy is a powerful tool for the analysis of substances in a variety of fields. For example, it has been applied to the detection of pathogens [65], explosives [7], transplant rejection [66], and cancer [4-6]. Although Raman spectroscopy can produce useful results for many applications, there are certain challenges that must be met.

One challenge is the translation of raw spectra into a final analysis or classification. For each individual spectrum, a number of preprocessing algorithms must be applied to reduce the effects of shot noise, cosmic rays, background fluorescence, and other distortions. Additional data reduction and feature extraction algorithms are often required to prepare the spectra for analysis and classification. The analysis itself can involve a multitude of statistical techniques, and the development of a classification model typically entails the application of complex learning algorithms.

Another challenge is the management of Raman spectra after they have been collected. Researchers at Wayne State University, like many others, have been collecting thousands of Raman spectra from hundreds of samples over a number of years. To analyze and compare spectra from different samples, it is necessary to form groups from the collected spectra that represent the types of samples that need to be analyzed. To accomplish this task, a procedure or tool for finding data that meets specified criteria in a spectral library must be utilized. In addition, the different groups have to be maintained throughout the processing and analysis of the data.

Currently, there appears to be little standardization among research groups with respect to how Raman spectra are treated. In general, each group develops custom implementations of the algorithms used in Raman processing and analysis. Many of the algorithms are very similar, so

having multiple implementations represents an unnecessary duplication of work. Furthermore, for those algorithms that are unique, it is more difficult to share them when other groups have custom processing pipelines.

3.4.1 Related Work

There are a variety of commercial software applications that can handle different stages of Raman spectral processing, analysis, and classification. Some commonly used applications that are customized for spectroscopic data include the GRAMS Suite (Thermo Scientific, Waltham, MA) and OPUS (Bruker Optics, Billerica, MA). These programs contain features for the acquisition, management, processing, analysis, visualization, and identification of spectra. In addition, more general statistical analysis packages are often used, such as SPSS (SPSS, Chicago, IL) or Origin (OriginLab, Northampton, MA). While most commercial applications are mature and contain many features, they typically lack the latest algorithms for tasks such as processing and classification. Their extensibility is also limited (or nonexistent), generally requiring the use of a proprietary programming language. Furthermore, the purchase of any commercial application will entail additional monetary cost.

If one wishes to develop his own Raman application, the use of a software development environment can greatly simplify the task. For example, MATLAB (The MathWorks, Natick, MA) is one of the most popular environments for Raman algorithm implementation. MATLAB provides a programming language with excellent support for numerical calculations and access to collections of functions for statistical analysis, signal processing, and other application areas. Another commonly used development environment is LabVIEW (National Instruments, Austin, TX). It uses a more graphical approach to programming with libraries for signal analysis and instrument control.

Other research groups have written papers describing their Raman processing software, most recently for the real-time analysis of data from *in vivo* probes. For instance, Bakker Schut et al. [15] described a real-time system that uses both GRAMS and MATLAB. Their software supports spectral preprocessing, analysis via least squares fitting, and classification via principal component analysis and linear discriminant analysis. In another paper, Motz et al. [14] presented a real-time system that utilizes LabVIEW and MATLAB. Their software implements spectral preprocessing, analysis via least squares fitting, and classification via logistic regression of the fit coefficients.

There are some related, free chemometrics software packages that aid in the analysis of spectroscopic data. The Biodata toolbox [67] for MATLAB provides a programming structure for managing spectroscopic signals and supports a number of algorithms for processing, analyzing, and classifying samples. It contains a few techniques (e.g., correlation analysis) that are not implemented in the system being presented, but it's also missing some advanced techniques of the system in areas such as processing and classification. Moreover, unlike the presented system, the Biodata toolbox is only useful for software developers because it requires a MATLAB installation and almost all functionality is accessed through programming commands.

ParLeS [68] is an application developed in LabVIEW for the analysis of spectra. It provides a variety of algorithms to process data and two classes of techniques for analyzing spectra and predicting variables. All features are accessed through a graphical interface that provides statistics to help assess the performance. Compared to the presented system, ParLeS is less capable in areas such as managing and classifying spectra. Unfortunately, the program no longer appears to be available.

There is a combination of attributes that together differentiate the software system presented in this chapter. First, the system is being released for free. The development version requires a copy of MATLAB, but the standalone version only needs the MATLAB Compiler Runtime to be installed. Second, the source code of the software is also being made available. This makes it possible to modify the system or extend it with new features. Moreover, any improvements can be shared with all users of the system.

A third benefit of the presented system is that it was designed to be easy to extend. It is based in the MATLAB environment, whose capabilities allow the implementation of almost any practicable feature. No compilation is required when adding or modifying code, and graphical debugging mechanisms are provided. This should also reduce the effort required to adopt new algorithms, which are often coded in MATLAB by their developers. Furthermore, it is easier to design and test new algorithms when the developer has a working system (i.e., the presented system) in which to build them.

A fourth benefit of the system is found in the diverse features that are included. For example, the software has powerful preprocessing capabilities, such as an enhanced version of an automated background subtraction algorithm [69]. In addition, the system allows spectra to be organized into groups, and it has the ability to search for spectra in a database. Moreover, the software enables the classification of spectra via techniques such as linear discriminant analysis, artificial neural networks, and support vector machines.

Lastly, despite its powerful techniques, the system is still relatively easy to use. All functionality is accessible through simple graphical interfaces; no programming ability is required. Furthermore, an effort has been made to automate the selection of algorithm parameters so that little adjustment is required from the user.

3.5 System Architecture

The presented system was built in the MATLAB environment for Windows (Microsoft, Redmond, WA), but it also works in MATLAB for Linux. MATLAB was chosen due to its inherent computational capabilities, suitability for mathematical algorithm development, and ease of extensibility. Moreover, MATLAB is a well-known program, and many algorithms for Raman spectra have already been written in its environment. For ease of distribution, the system can be compiled to run on computers without MATLAB.

The system is implemented in the form of a menu-based graphical user interface (GUI), which is shown in Figure 3.1. Each menu item executes a function that operates upon data loaded in the system (e.g., Raman spectra). This type of GUI was chosen because it is familiar to most users. In addition, it is easy to add new functions to the system by adding new menu items.

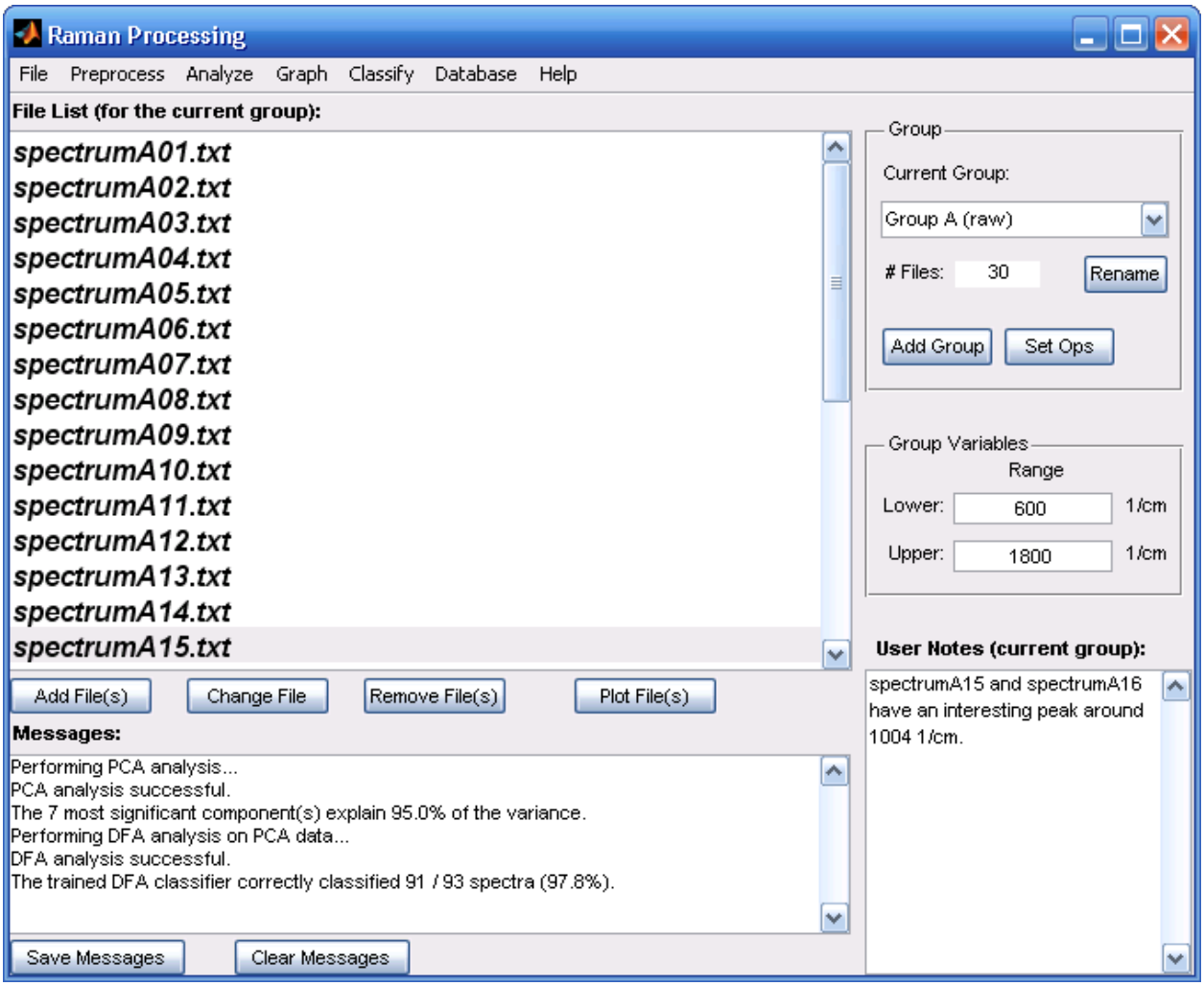

Figure 3.1: The primary graphical user interface of the developed Raman processing system. The file list (top left) shows the files contained in the currently selected group (top right).

The system's menu items and thus its primary features are organized into a series of categories. The categories include file loading/saving, preprocessing, analysis, graphing, and classification. Significant features that are implemented under each of these categories will be described in later sections of this chapter. The typical flow of data through the features of the system is shown in Figure 3.2.

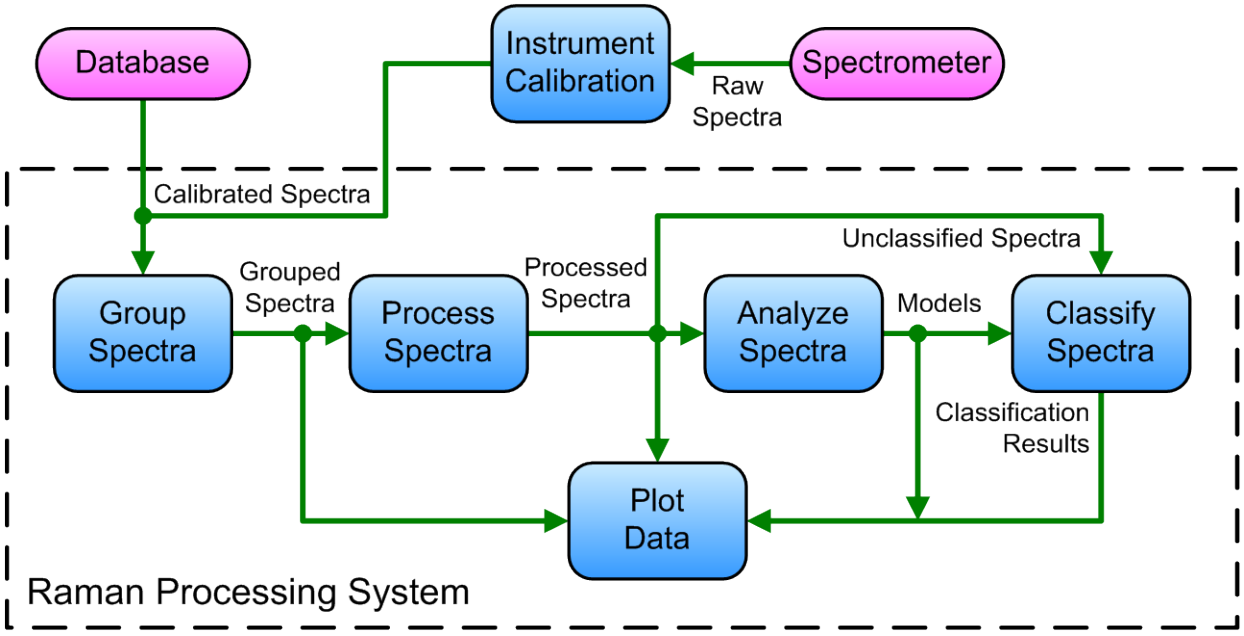

Figure 3.2: Diagram of the flow of data through the system.

The remainder of the GUI is divided into several sections (Figure 3.1). In the top left area, the list of files in the currently selected group of spectra is shown. In the bottom left corner, status messages are displayed when executing various functions. In the top right area, the user is allowed to select and manipulate the groups of spectra. Finally, in the bottom right corner, there is a box in which the user can enter any notes.

3.5.1 Data Structure and Extensibility

The system's current data is stored in the fields of a single structure (a type of computer programming variable). The fields are numerous and contain the complete state and parameters of the system. For example, some of the fields are used to hold groups of spectra, including the list of files associated with each group. Other fields store the information necessary for various analysis and classification models, such as the weights for a trained artificial neural network. The main data structure and its fields are accessible to any function in the system.

The primary features of the system are accessed through the menu of the GUI. When the user clicks on a menu item, an associated function is passed a copy of the main data structure and executed. The function is then free to access the data, run calculations, display an output, or perform any other necessary task.

The system was designed to make it as simple as possible to add a new feature, such as a new classification algorithm or analysis technique. One only has to create a new menu item and write an associated callback function. The challenging steps are writing the custom algorithm for the desired task and linking it with the internal structure of the system. To ease the burden on the developer, a large number of utility functions are included with the system. For example, there are utility functions for loading spectral data from files, displaying figures, training classifiers, and more. If a new feature requires additional fields in the main data structure, they can be added easily.

3.5.2 Data Input and Management

The system presumes that all spectral data has been previously collected and calibrated for typical wavenumber and intensity variations caused by the equipment. In addition, the spectral data needs to be stored in simple text files (with one column of wavenumbers and another column of corresponding intensities). For better compatibility with existing data sets, the system can convert files from the commonly used Thermo Scientific SPC file format (Thermo Scientific, Waltham, MA). Once the data is available, the system allows the user to input text files by selecting them individually, by filtering the files in a folder, or by using the files stored in multiple folders.

One of the key uses of Raman spectroscopy is to collect data from various sets or groups of samples to determine the similarities or differences among them. For this reason, the system has

been structured around the notion of groups of spectra. The user creates groups that represent different types of samples, such as “healthy” or “cancerous”. Each spectrum that is inputted into the system must be assigned to one of the groups. In addition, if the spectrum files are organized into separate folders, groups can be created automatically from the folder structure.

This organization of data enables streamlined processing and analysis of spectra. For example, it allows similar spectra (those within a group) to be processed with the same parameters. The organization also makes it easy to compare different groups of spectra by plotting their mean spectra or performing statistical analyses on them. To help classify new spectra, learning algorithms can be trained to distinguish among groups of known spectra.

Once groups have been formed, it may be useful to create new groups based on the existing groups. For instance, one might wish to combine two similar groups or split a diverse group into multiple groups. To enable this kind of task, the system supports set operations (union, intersection, and difference) that generate new groups from existing ones. A simple interface allows the user to repeatedly perform these operations and preview the results before committing them to a new group.

After the desired groups have been made, the system allows the group information to be stored in a file for later retrieval. This “group file” includes the path of each spectrum file and the wavenumber range for each group. In addition, the system saves any notes the user has entered for each group.

3.5.2.1 Database Access

A database can be very useful in the management of a large collection of Raman spectra. It enables one to store any kind of information about the spectra, such as the date of collection, type of specimen, and collection parameters. Later, the database can be queried to return spectra

that match some desired parameters. For example, one could search for spectra of healthy and cancerous tissues from the same organ in order to help develop a model for diagnosis.

The system has the ability to connect to SQL Server or an Access database (Microsoft, Redmond, WA) when running in Windows. With simple modifications, it should also be possible to communicate with other database systems that have an OLE DB (Microsoft, Redmond, WA) provider for Windows. Once a connection is made, the database can be queried or updated using Structured Query Language (SQL) commands.

Since there are countless ways to set up a database and types of information to store, each research group will have to develop a custom interface for their particular configuration. A simple example using an Access database is provided. The MATLAB GUI design environment (GUIDE) can be used to develop a user interface, and SQL commands must be written to access a particular database structure. Once spectra have been located through a database query, they can be automatically copied to a local drive and added to a group. From that point, the spectra can be processed and analyzed like any other spectra in the system.

3.5.3 Spectral Processing

The raw data is initially processed by a median filter to remove spikes due to cosmic rays and reduce noise in each spectrum. Next, a technique based on wavelet (Symlet) transformation and coefficient thresholding [70] is used to further reduce shot noise in each spectrum. The system then employs the "adaptive minmax" method [69] to estimate and subtract the background fluorescence from each spectrum. This method uses several modified polynomial fits to automatically approximate the background fluorescence without any user interaction.

Some new features have been added by Alex Cao to enhance the background fluorescence subtraction. First, the algorithm weights certain points of the spectrum to reduce the likelihood

of the estimated background having a higher intensity than the spectrum. Second, the user is given the option to perform additional background subtraction if he is not satisfied with the original approximation. Using a plot of the mean spectrum of a group of processed spectra, he can select a range where the background subtraction will be performed again. For example, this could correspond to a region where the estimated background had a significantly lower intensity than the spectrum. The algorithm will then process each spectrum within the selected range and return a new spectrum with the revised background subtracted.

Next, each spectrum's intensities are normalized to the range of 0–1 arbitrary units by subtracting the minimum value and dividing by the maximum value. This reduces differences in overall intensity among spectra that can occur due to changes during the collection process (in probe orientation, power level, magnification, etc.). However, the transformation preserves the relative intensities of peaks within each spectrum. Having the intensity values normalized to a common range can provide better performance with some algorithms, such as a neural network classifier.

Optionally, an artifact removal algorithm can be executed while processing the spectra. Some hardware configurations may result in an unwanted signal being added to each spectrum as the light travels through the optical paths from and to the spectrometer. For example, a sapphire lens may introduce sapphire-related peaks into a spectrum. To remove them, the user must first obtain a spectrum containing only the unwanted artifacts. One way to acquire this is by scanning aluminum [14], whose low response is often insignificant compared to the artifacts of the optical paths. The artifact spectrum is scaled to best fit each spectrum being processed and subtracted. Figure 3.3 shows the results of applying the artifact removal algorithm to remove peaks associated with a sapphire ball lens.

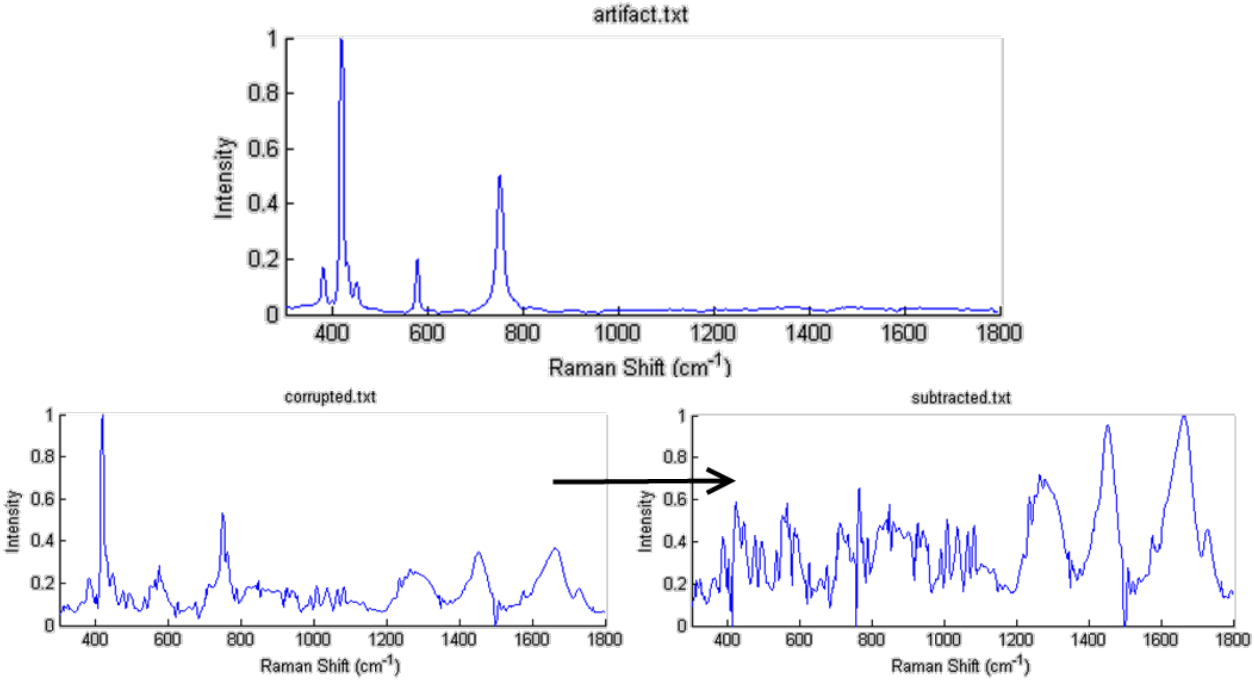

Figure 3.3: Example of the artifact removal algorithm. The top spectrum represents the artifacts from a sapphire balls lens. The left plot shows an initial spectrum with artifacts, and the right plot displays the spectrum after artifact removal.

3.5.3.1 2D Morphological Background Subtraction

Preliminary work was done on a new technique to remove background fluorescence from Raman spectra. It uses mathematical morphology to detect the background fluorescence that should be removed. This is an extension of the work described in [71] and [72], which used 1-dimensional (1D) morphological operations. Here, 2-dimensional (2D) operations were used to attempt better removal of curved backgrounds.

First, the plot of a Raman spectrum is divided into a 2D grid. Then the morphological operation known as "closing" [73] is used to fill in the spaces under the peaks of the Raman spectrum. It does this by moving an entity known as a structuring element through the grid and performing simple morphological operations. For this work, the structuring element was a circle whose size can vary according to the spectra being analyzed. For example, the size of the circle can be set according to the fluorescence-to-signal ratio of [69].

A similar technique called the "rolling-circle filter" developed in [74]. However, it is not based on mathematical morphology. Morphology allows the shape of the structuring element to be easily modified, which may allow it work better on different types of spectra. For more details on the subject of mathematical morphology, the reader may consult [73].

The following figure shows the difference between the adaptive minmax technique and the developed 2D morphological technique for a spectrum from a pear. The new technique follows the spectrum closely and avoids cutting into the spectrum. It also does a reasonable job of following the spectrum at its ends. The decision of which technique is better is the subject of future research.

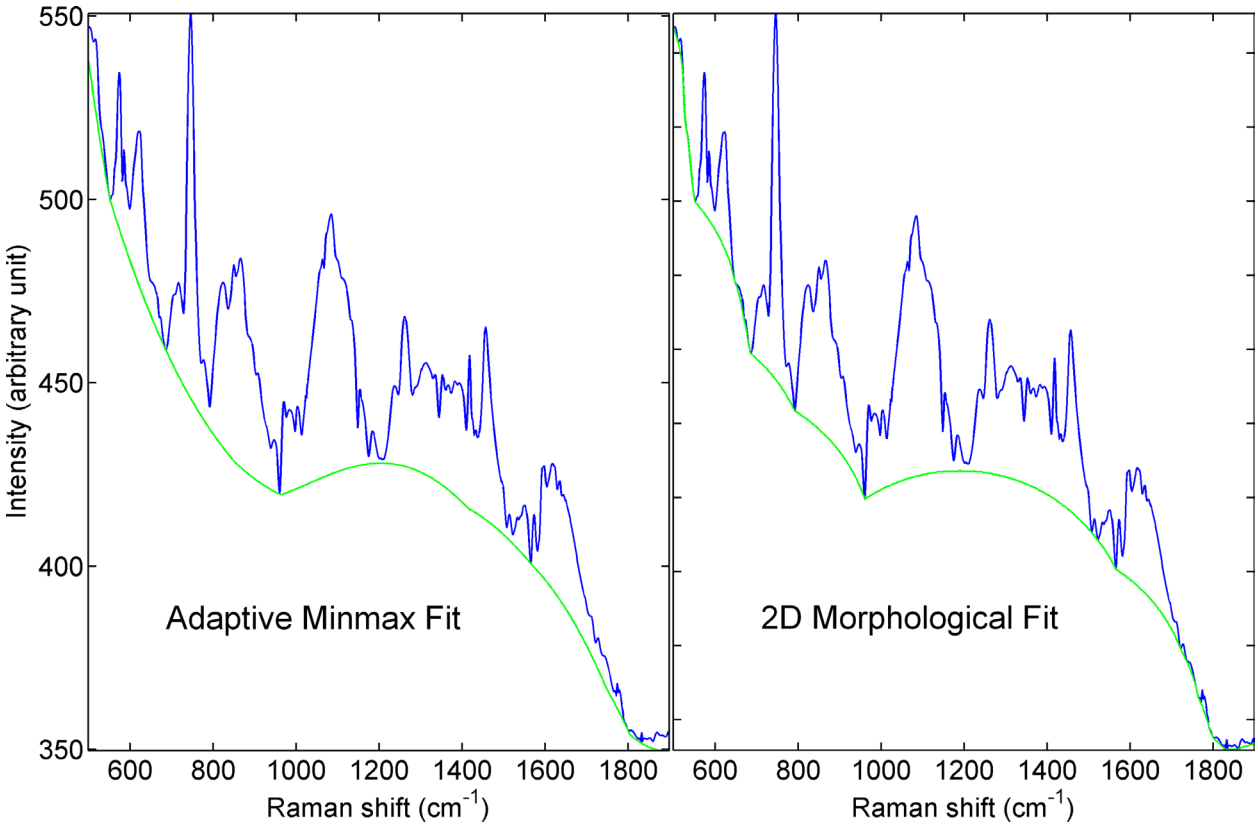

Figure 3.4: Comparison of the fitted backgrounds (green) for the adaptive minmax and 2D morphological background subtraction techniques on a pear spectrum (blue).

3.5.4 Spectral Analysis and Classification

The presented system currently implements several analysis and classification techniques, such as principal component analysis and support vector machines. For each of these techniques, a model is created by training the algorithm with selected groups of spectra. The model can then be applied to other groups of spectra for the purpose of extracting features or assigning classifications. The system allows models to be saved to and restored from a file so any results can be reproduced easily in the future. To support diverse spectrum files, all spectra are automatically binned to a common wavenumber range and resolution when generating or applying a model.

A Raman spectrum typically consists of hundreds of data points, many of which may be significantly correlated with each other when considering one or more groups of spectra. To reduce this redundant information and highlight important features, the system implements a technique called principal component analysis (PCA). PCA is a statistical technique that transforms the spectral data into new, uncorrelated variables (principal components) based on the variations in the data [49]. Because only a few principal components are necessary to represent the majority of the total variance in the data set, PCA is an excellent way to reduce the dimensionality of the data. The system utilizes only the most significant components that account for at least 95% of the variance in the spectra being analyzed. Consequently, each spectrum can be represented in terms of the principal component variables as a small set of values, called scores.

The system currently supports three different algorithms for spectral classification. The inputs to these algorithms can be either principal component scores (produced by applying a PCA model) or actual spectral intensity values. As mentioned above, the algorithms are trained

with selected groups of spectra to produce models. These models can be applied to assign a classification (e.g., healthy or cancerous) to groups of unknown spectra. The results are displayed in the program's "Messages" box and can be summarized in a pie chart or saved to a tab-delimited text file.

Alternatively, a leave-one-out classification can be executed using any of the three implemented techniques. A leave-one-out classification evaluates the system's classification performance by estimating its ability to identify unknown spectra using all of the known spectra. For this process, the user selects two or more groups of spectra that the system should ideally be able to differentiate. The system then computes PCA scores and trains a classification model using all of the spectra except one from the selected groups. Next, the classification model is applied to the excluded spectrum to determine if the system is able to correctly associate it with its source group. This process is repeated until every spectrum has had the opportunity to be the excluded one. The results for each spectrum (and each group) are shown in the "Messages" box. The results can also be summarized in pie charts (Figure 3.5) or saved to a tab-delimited text file for further analysis.

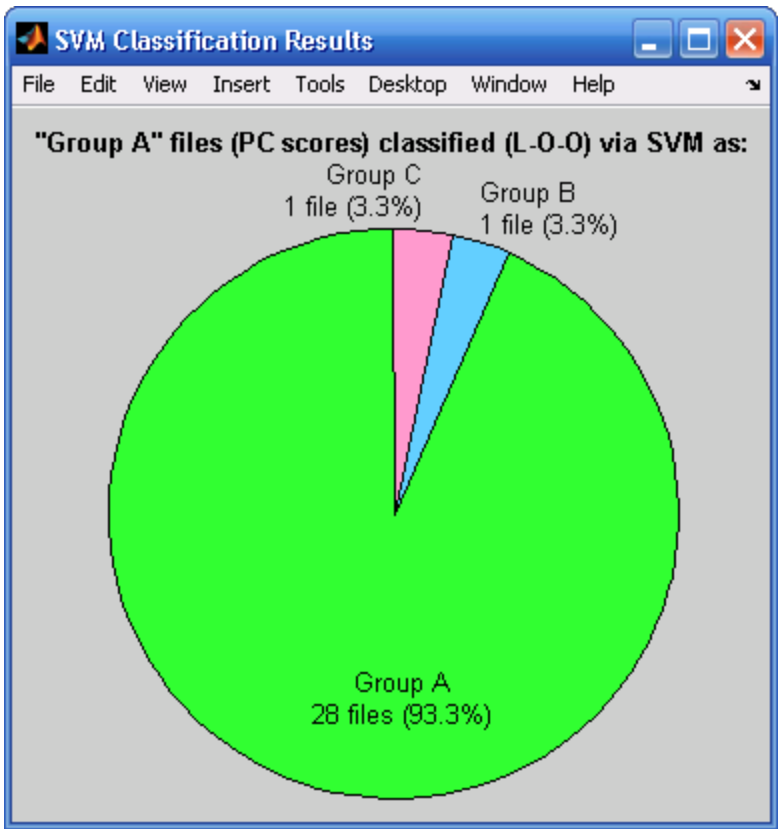

Figure 3.5: A pie chart generated by the system that shows the results of a leave-one-out (L-O-O) evaluation of a support vector machine classifier.

3.5.4.1 Classification Algorithms

The first implemented classification algorithm is based on linear discriminant analysis (LDA). LDA is a statistical technique that is used to find the variables (e.g., points in a spectrum) that best discriminate between two or more groups [59]. It attempts to find linear combinations (called discriminant functions) of the input variables that maximize between-group variation while minimizing within-group variation. Discriminant function scores are calculated by multiplying each discriminant function's coefficients with the corresponding points in the spectrum. After finding the discriminant functions, the information can be used produce a different classification function for each group. For each spectrum, a classification score is calculated for each group, and the spectrum is assigned to the group with the highest score.

The second classification algorithm is implemented using an artificial neural network (ANN). ANNs are a class of learning techniques that are based on biological neural networks [60]. A typical ANN consists of interconnected layers of processing units (called neurons) that produce outputs based on a weighted function of their inputs. Training an ANN model is an iterative process that can be relatively slow for large amounts of sample data. On the other hand, a model can be applied to classify a new spectrum rather quickly. An ANN classifier can be very accurate if the network is structured and trained properly and if the new data is sufficiently similar to the training data.

The third classification algorithm employs support vector machines (SVMs). SVMs are another class of learning techniques that use hyperplanes to separate data in a high-dimensional space [61]. By using a nonlinear transformation and multiple SVMs, it is possible to classify complex spectra into one of multiple groups. Training a SVM model can be relatively slow, but applying the model to new spectra is much faster. A SVM classifier can attain high accuracy and can be easier to train than an ANN.

3.5.4.2 Peak Characterization

Preliminary work was done on a peak characterization technique to fit a Gaussian–Lorentzian sum model to the peaks in a spectrum. The choice of the model was motivated in part by [75], which discussed how the peaks in Raman spectra often have a shape that can be approximated by a Gaussian–Lorentzian sum model. The following equation represents the model.

$$f(x,a,c,w,r) = ra\mathrm{e}^{-(\ln 2)\left(\frac{x-c}{w}\right)^2} + (1-r)\frac{a}{1+\left(\frac{x-c}{w}\right)^2} \qquad \text{Equation 3.1}$$

In Equation 3.1, the following terms are used:

- $f(x, a, c, w, r)$ is a function that generates the intensity values of the modeled peak
- x represents the current wavenumber, usually expressed in cm^{-1}
- a represents the amplitude of the Gaussian–Lorentzian sum model
- c is the center wavenumber of the Gaussian–Lorentzian sum model
- w represents the width of the Gaussian–Lorentzian sum model
- r represents the ratio of the Gaussian model contribution to the Lorentzian model contribution for the combined (summed) model

To begin, the general regions of the peaks in a spectrum must be identified. This is accomplished via a simple algorithm. First, a sliding window of a chosen width is moved across the spectrum. For example, a width of $9\ \text{cm}^{-1}$ was used in the following example. The wavenumber of the highest intensity value within the window is recorded when the window is at a specific location.

As the window moves, the location of the highest intensity will change to different wavenumbers. Each time there is a change, the wavenumber is recorded. This process continues until the window has gone through the entire spectrum. The resulting list of wavenumbers represents the points that are highest among their neighbors, which are likely to be the peaks.

Next, an approximate width for each peak must be obtained. To determine this, the algorithm starts at each identified wavenumber (peak) and inspects the intensities of the points to the left side. As the algorithm moves outwards from the starting point, it checks to ensure that each point is less intense than the last (representing a downward slope). Once a point is reached where the intensity goes up, the algorithm stops and records this as the left end (edge) of the peak.

The process is repeated with the algorithm moving from the center towards the right side of the peak. Thus, the algorithm stops when the slope changes from downwards to upwards and records the point as the right end of the peak. Finally, the width of the left and right sides are clipped to the smaller of the two. This helps ensure that the approximate peaks aren't too wide.

After that, the algorithm eliminates from consideration any peaks whose heights are too small. The height is defined as the difference from the tip of the peak to the lowest point in the identified region of the peak. For the following example, the any peaks with a height less than 1% of the highest intensity in the spectrum were eliminated.

At this point, a set of peaks (with roughly identified center points and widths) is available. Now each peak is fit to the Gaussian–Lorentzian sum model of Equation 3.1. To accomplish this, a method is needed to find the model coefficients.

An objective function was created that computes the coefficient of determination (R^2) of the fitted model with respect to the original intensity values of a peak. The model parameters that minimize this function are found via a standard optimization technique, the Nelder–Mead simplex method [76]. Thus, the parameters (a, c, w, r) that provide the best fit of the model to the actual intensity values of the peak are determined. This process is repeated for all the identified peaks.

An example of the output of running the peak characterization technique on a Raman spectrum is shown in Figure 3.6. The models have reasonably good fits to the underlying Raman data. There is some deviation where the peaks were skewed from a true Gaussian–Lorentzian shape. A skew parameter and a more accurate method to detect the initial widths of the peaks would improve the output in some cases.

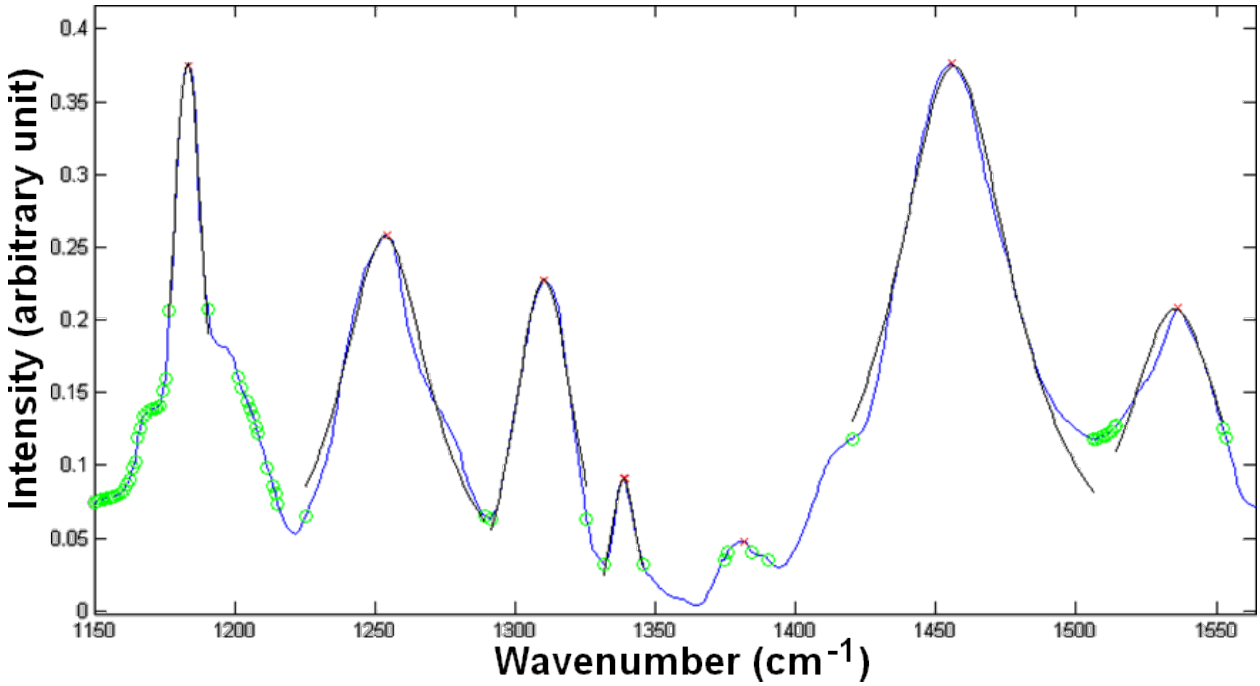

Figure 3.6: Image showing the models (black lines) that were fit to the peaks of a Raman spectrum. The red asterisks indicate the initial approximations of the peak centers, and the green circles denote the ends of regions initially considered to be peaks.

3.5.4.3 Discriminating Peak Analysis

Preliminary work was done on a technique to find peaks that best discriminate between two groups of spectra. Using partial least squares regression (as described in section 2.2.2.1), the discriminating regions between two groups of spectra can be obtained. To achieve this discrimination, a binary response variable is created with a value of '0' for all the spectra from the first group and a value of '1' for all the spectra from the second group. The intensity values of the spectra are treated as predictor variables.

Partial least squares regression is then used to find the coefficients of the relationship between the response and predictor variables. The largest coefficients should be those corresponding to the most discriminating regions of the spectra. Thus, the coefficients that exceed a chosen threshold are used to represent the most discriminating regions between the two groups of spectra.

By combining this with peak characterization (section 3.5.4.2), the most significant peaks in a spectrum can be found. In addition, using a peak table like the one in [77], the peaks can be associated with specific chemical compounds. This provides a basic tool for the assisted chemical analysis of Raman spectra. The following figure shows the results of running the technique on a Raman spectrum.

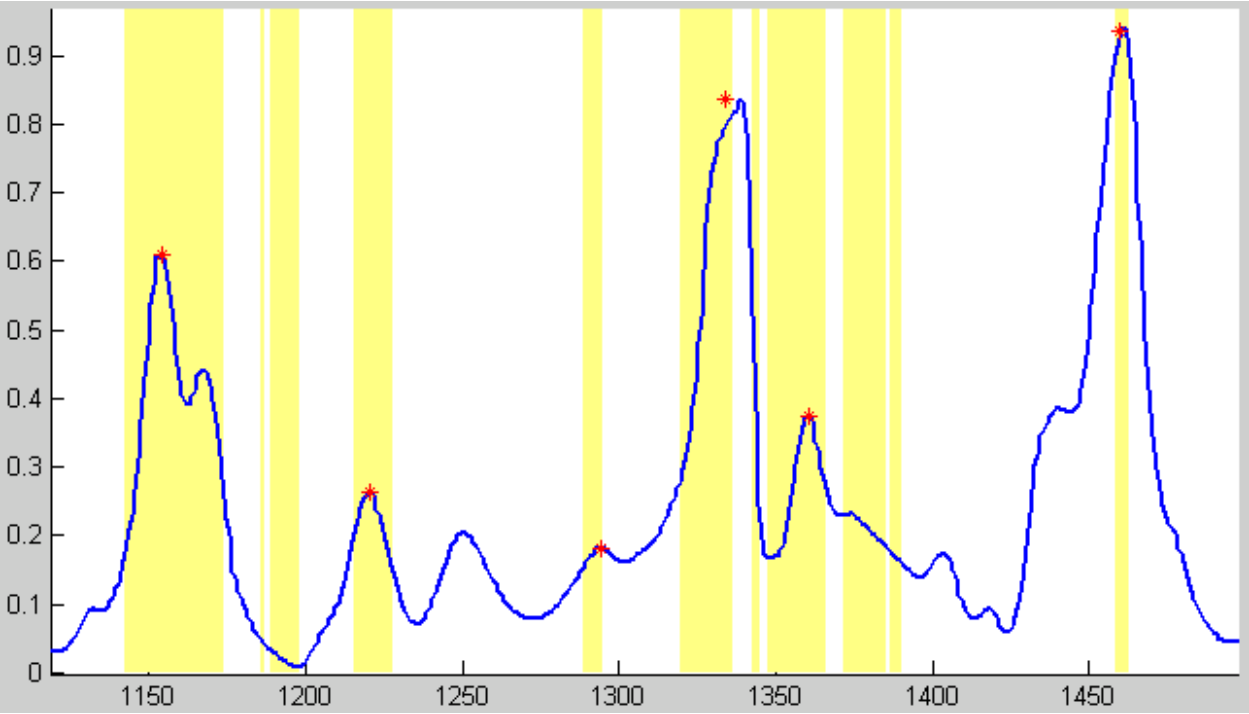

Figure 3.7: Image showing the execution of the discriminating peak analysis technique. The yellow areas indicate wavenumber regions that best discriminate between two groups of spectra, and the asterisks represent the identified peaks of chemical components in the discriminating regions.

3.5.5 Spectral Graphing

The system has a variety of features to enable the visualization of data. Individual spectra can be plotted as well as the mean spectra of groups. In addition, it is possible to plot the standard deviation or standard error of the mean spectra or the difference between mean spectra. Any of these plots can be shown alone or combined with other plots in a single figure (Figure 3.8).

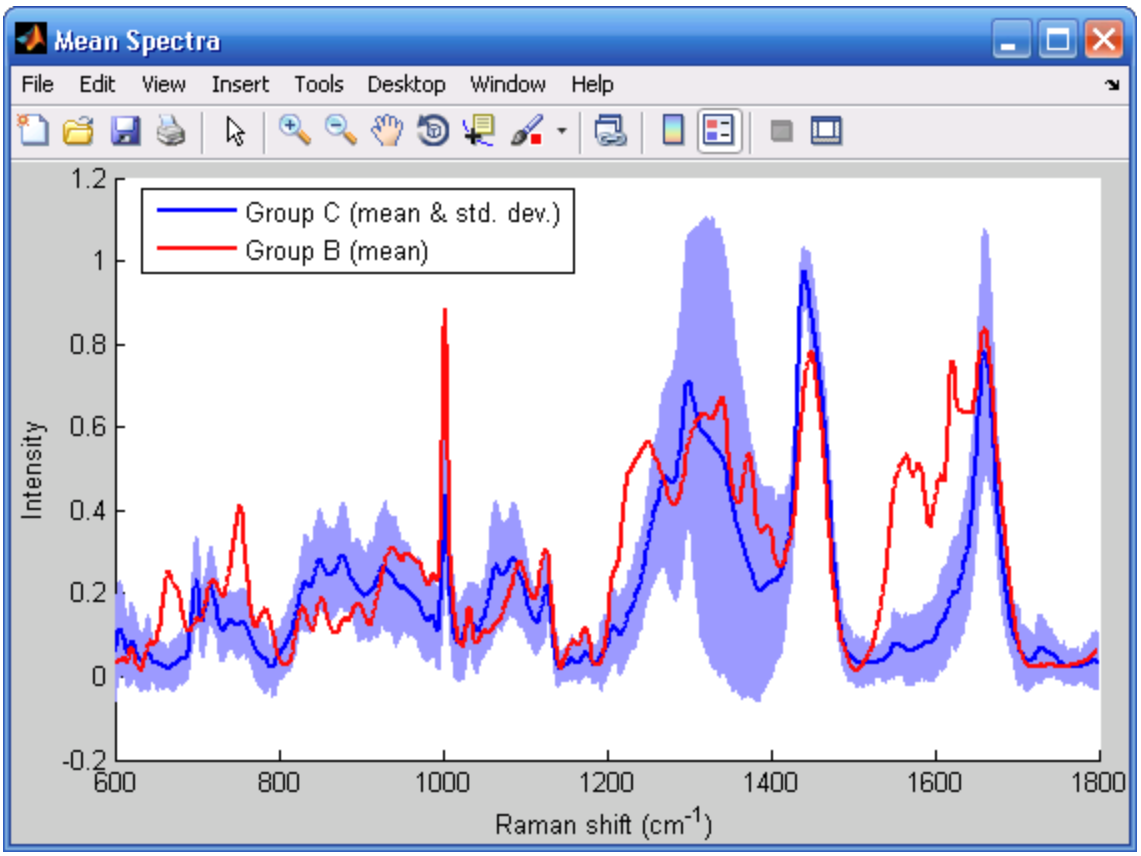

Figure 3.8: A figure generated by the system that shows plots of the mean spectra and an interval of ±1.96 standard deviations for two groups of spectra.

The system is also able to plot various parameters derived during spectral analyses. The coefficients for each principal component of a PCA or for each classification function of a LDA can be graphed to provide a visual representation of the models. In addition, the principal component scores or discriminant function scores of spectra can be plotted with customizable settings for which scores are displayed along which axes (Figure 3.9). This enables the user to see how well the spectra are separated in principal component space or discriminant function space.

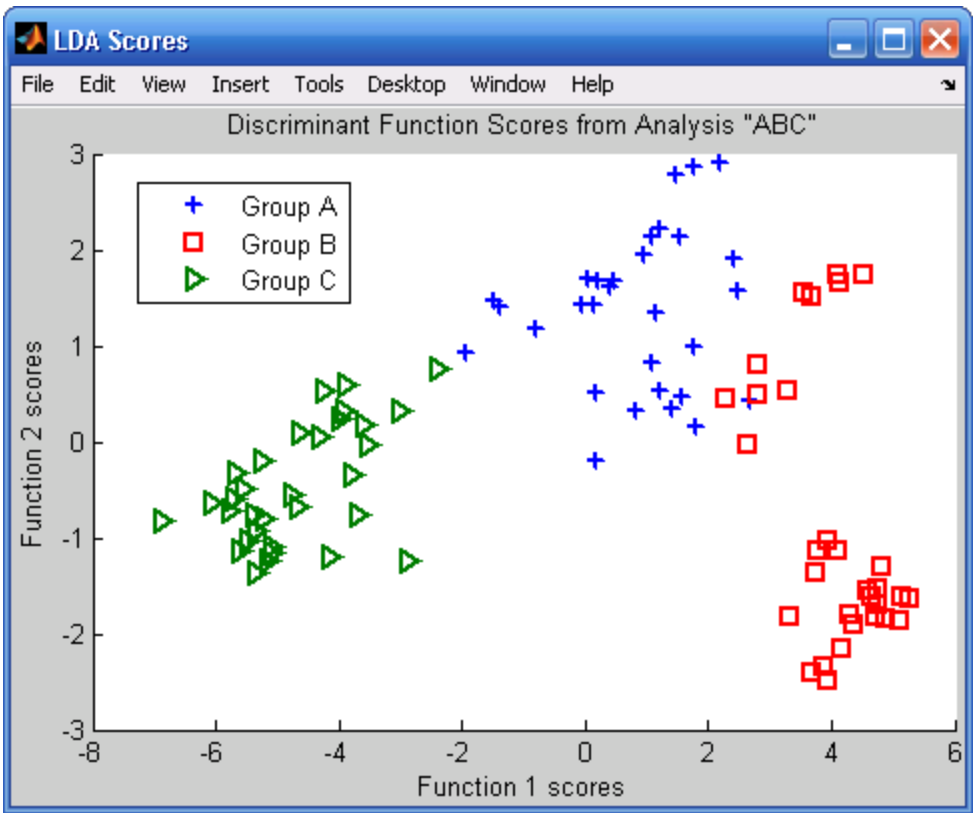

Figure 3.9: A figure generated by the system that shows a plot of the discriminant function scores of three groups of spectra.

Peak lookup is another useful feature that can be applied to the plot of any spectrum. It places a cursor on the plot that can be manually positioned at peaks or other points of interest. Using a table, the system displays the name of any biological components (such as proteins or lipids) whose spectra have a significant peak at the cursor's location. This feature can aid in understanding the composition of a material based on its Raman spectrum. The table of component peaks can be customized as needed to fit the desired application.

3.6 System Evaluation

To demonstrate its speed and effectiveness, the system was used to process an example data set on a computer with a 3.4 GHz Pentium D (Intel, Santa Clara, CA) processor. The data set consisted of spectra collected from biological specimens with an inVia Raman microscope

(Renishaw, Wotton-under-Edge, UK) using a 785 nm laser and two different lenses. The spectra represented three different groups of similar specimens. Since the purpose of this chapter is to introduce the system rather than analyze a particular data set, the specifics of the specimens are not considered to be relevant. Thus, the three groups of specimens will be referred to as Group A, Group B, and Group C. The groups contained 30, 31, and 32 spectra, respectively.

Figure 3.10 shows the mean spectra of the unprocessed data from Groups A, B, and C. Since the background has not yet been subtracted, the relative heights of the lines in the figure are not significant. Some common peaks can be seen among the groups, but it is hard to see the exact differences without further processing. Figure 3.8 in the previous section shows a view of the processed spectra from groups B and C, which makes it much easier to identify the differences.

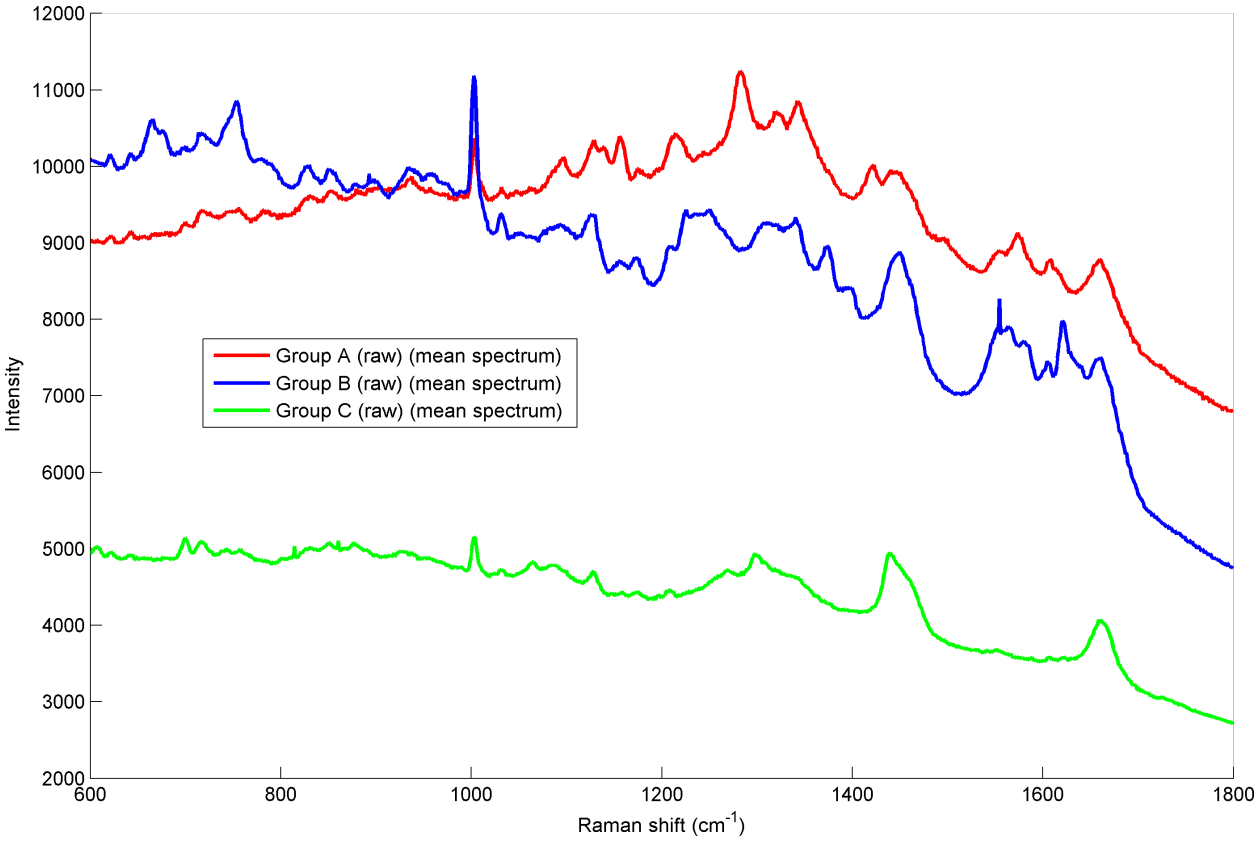

Figure 3.10: Mean spectra of the unprocessed data for the three groups of the first data set.

The system was first used to carry out the initial processing of the raw spectra. It took 66.98 seconds for the system to filter, subtract the background from, and normalize all 93 spectra (each consisting of 1310 data points). Afterwards, the spectra were binned to a common wavenumber axis and used to generate a PCA model. This process took 0.64 seconds and produced a model that used the seven most significant principal components.

Next, a series of tasks involving classification were performed to evaluate the system's performance. The execution times for these tasks are summarized in Table 3.1. The first task was to train classifier models (LDA, ANN, and SVM) using the PCA scores of the 93 preprocessed spectra. The second task was to process a single new spectrum (not one of the original 93 spectra) and apply each of the models to classify it. All three models classified the spectrum correctly. The final task was the leave-one-out evaluation of each classification algorithm using the 93 preprocessed spectra. The final task's time covers 93 iterations of PCA computation, model training, and spectrum classification.

Classifier	Model Training	New Spectrum Classification	Leave-one-out Classification
Linear Discriminant Analysis	0.016 s	0.795 s	7.5 s
Artificial Neural Network	0.805 s	0.797 s	85.8 s
Support Vector Machine	1.350 s	0.787 s	134.7 s

Table 3.1: Execution times required by the system to train a classifier model (using precomputed principal component scores), classify a new spectrum (including processing time), and perform a complete leave-one-out evaluation (using preprocessed data). A total of 93 spectra divided into three groups were used for the training and leave-one-out classification.

The leave-one-out evaluation demonstrated that the system had relatively good classification performance on the example data set. The classification accuracies are presented in Table 3.2, and one of the generated pie charts is shown in Figure 3.5. All of the classifiers were over 90% accurate for each group, corresponding to misclassifications of at most three spectra per group.

Classifier	Group A Accuracy	Group B Accuracy	Group C Accuracy
Linear Discriminant Analysis	93.3%	90.3%	100%
Artificial Neural Network	93.3%	100%	100%
Support Vector Machine	93.3%	96.8%	93.8%

Table 3.2: Classification accuracies of the system for the three example groups determined through a leave-one-out evaluation of each classifier technique.

A second data set consisting of fruit specimens (apple, mango, and pear) was collected with a VERAX spectrometer (InPhotonics, Norwood, MA) solely for the purpose of testing the system. This is a lower-resolution device that uses a fiber optic probe. One hundred spectra were captured from each of the ripe fruit specimens. To help classify the decay of the fruit, 100 additional spectra were collected from the mango after a short period of time (2 days) and again after a longer duration (17 days). The data set was captured quickly in a less controlled environment (in terms of lighting, temperature, etc.), increasing the variance and thus the challenge for the data analysis.

Two more leave-one-out evaluations of the system's classification algorithms were performed using the fruit data. The first evaluation (Table 3.3) used the 100 spectra from each ripe fruit, and the second evaluation (Table 3.4) used the sets of 100 mango spectra from days 0, 2, and 17. The results show that the system's classifiers were able to achieve high accuracies (94–100%) with the fruit data.

Classifier	Apple Accuracy	Mango Accuracy	Pear Accuracy
Linear Discriminant Analysis	95%	100%	99%
Artificial Neural Network	94%	100%	97%
Support Vector Machine	96%	100%	96%

Table 3.3: Classification accuracies of the system for three fruits determined through a leave-one-out evaluation of each classifier technique.

Classifier	Day 0 Accuracy	Day 2 Accuracy	Day 17 Accuracy
Linear Discriminant Analysis	100%	95%	95%
Artificial Neural Network	100%	94%	96%
Support Vector Machine	100%	95%	96%

Table 3.4: Classification accuracies of the system for a mango on three different days determined through a leave-one-out evaluation of each classifier technique.

The system's processing algorithms (noise filtering, background subtraction, and normalization) were previously used by Wayne State University researchers (before the GUI and other algorithms were developed) to help classify tissues. Other software was used for feature extraction (PCA) and classification (LDA), but those techniques (and others) have since been implemented in the system. High classification accuracies were attained for a number of tissue types. For example, sensitivities of 91–100% and specificities of 88–100% were achieved when detecting pancreatic cancer [4], neural crest tumors [5], and kidney tumors [6]. These results provide further evidence of the efficacy of the system.

3.7 Discussion

The computational treatment of Raman spectra is a complex process with no standard implementation of commonly executed methods. Therefore, a freely available system has been developed that handles the processing, analysis, and classification of Raman spectra. The system includes many powerful algorithms that are easily accessible through a simple user interface. In addition, the system is open source and extensible, allowing the community of Raman researchers to make continual improvements to the software. The developed system was shown to be fast and accurate in the analysis of two test data sets.

The claim that the system represents an optimal collection of algorithms is not being made. There are numerous areas that could be improved, such as the cosmic spike filter, spectral feature extraction, and the execution speed of some of the algorithms. However, this is one of the

primary motivations for sharing the work. If others contribute to the development, weaknesses can be eliminated more quickly, and everyone benefits from the work. It is intended that the system will grow to become a powerful tool for any application of Raman spectroscopy.

3.7.1 Future Work

Work is continuing on various aspects of the system. New algorithms are being developed for processing and analysis, and bugs found by users are being fixed. Interoperability is going to be improved by supporting the import and export of data in formats compatible with other programs, such as SPSS. Moreover, it is hoped that other researchers will contribute to the system so their enhancements can be integrated with the main distribution.

One of the areas being worked on is better integration with the process of collecting Raman spectra. Seamlessly reading the output of Raman acquisition software or communicating directly with a spectrometer is necessary for real-time analysis and classification of spectra. An interface has already started being coded for a VERAX spectrometer with a probe suitable for use *in vivo*. To make this practical, additional calibration algorithms typically employed by Raman acquisition software will need to be implemented.

Some preliminary work has also been done on automating the system. This will allow a number of processing and analysis steps to be executed without user interaction. Combined with the ability to collect spectra, this will enable a variety of real-time *in vivo* applications. For example, the system may be used to help integrate Raman spectroscopy with image-guided [20] and minimally invasive [19] surgery. This work is started in the following chapter.

Chapter 4:
Image Guidance and Robotics for Sensors

4.1 Introduction

The usefulness of a sensor can be increased by combining it with supporting systems. For example, Chapter 3 presented a software system that processes sensor data to extract meaningful information. Another type of system that can aid in the use of an analytical sensor for real-time applications is an image guidance system. An image guidance system can be used to accurately track the position of a sensor relative to its environment, and it provides an intuitive means through which collected and processed data can be presented to a user. When such a system is used for surgical procedures, it is known as an image-guided surgery system.

As explained in Chapter 1, a Raman spectrometer was chosen to demonstrate the sensor-related concepts of this dissertation. Therefore, this chapter describes an image-guided surgery system that is integrated with a Raman sensor. The system is an updated version of the one presented in [20] and builds on work from [78].

The features of the presented system are not just applicable to Raman sensors or surgical procedures. The ability to track a sensing device and display derived information on imaging data could be useful for many types of sensors in various environments. Therefore, the work described here could be applied to other kinds of sensors and applications.

4.2 Overview

In this chapter, the integration of a Raman spectroscopy-based sensor with an image-guided surgery system is presented. Such a system could provide a surgeon with both a diagnosis of tissue being analyzed (e.g., cancerous or normal) and location information displayed within an

imaging modality of choice. This kind of rich information could lead to faster diagnoses and enable more accurate treatments.

A test platform consisting of a portable Raman probe attached to a passively articulated mechanical arm was used to scan and classify objects within a phantom skull. The prototype system was successfully able to track the Raman probe, classify objects within the phantom skull, and display the classifications on medical imaging data within virtual reality and augmented reality environments. With further research and improvements to the system, its utility can be enhanced, and it could further the field of sensor-based computer-assisted surgery.

4.3 Background and Motivation

Current techniques in image-guided surgery rely primarily on visual feedback from the surgical site. This chapter addresses the issue of extending this feedback by adding a sensing modality, Raman spectroscopy, to the already successful techniques of image guidance. It is hypothesized that other modalities of information from the surgical/tumor site based on these non-visual (biochemical) aspects will enhance the surgeon's ability to more completely define resection margins.

Conventional histopathology lacks both the capability for providing immediate feedback and the precision to quantify the extent of disease, particularly in the early stages. Final results usually require 12–24 hours. Even the examination of the more immediate frozen sections typically requires at least 20 minutes from the time the tissue is removed until the time an answer is available. During tumor-removal surgeries (e.g., for brain cancer), this means the patient must remain open for a longer operative time.

Raman spectroscopy is a technique capable of detecting normal and abnormal regions of tissue [40]. Its near-real-time analysis and the fact that it does not require sample preparation

make it highly suited for *in vivo* applications [13, 55]. Image-guided surgery helps the surgeon position and track instruments (such as a Raman probe) inside the body [79], making it a natural complement for Raman spectroscopy. Integration of this sensing technology with image-guided surgery should help maximize its usefulness for *in vivo* applications. Thus, this chapter investigates the integration of a Raman probe with an image-guided surgery system for future applications like the diagnosis of cancer.

4.3.1 Raman Spectroscopy

Raman spectroscopy is a near-real-time technique that measures the wavelength and intensity of light inelastically scattered from molecules. In Raman spectroscopy, a specimen is irradiated with a laser light, resulting in the scattering of light due to its interaction with the vibrating molecules in the sample. The majority of scattered light has unchanged frequencies (Rayleigh scattering), whereas the rest is shifted in frequency (Raman scattering) by an amount characteristic of the frequency of the vibrating molecules. These vibrations are a function of molecular conformation, the distribution of electrons in the chemical bonds, and the molecular environment. Disease leads to changes in the molecular composition and morphologic appearance of affected tissues. Since Raman spectroscopy is sensitive enough to detect these molecular changes, it is a logical choice for the diagnosis of cancer [17, 25, 55, 80-85]. This technique is also well-suited for *in vivo* applications because it is non-destructive and requires no sample preparation or contrast-enhancing agents. These features make it appealing for real-time medical diagnosis.

Using recent developments in near-infrared Raman spectroscopy, *in vivo* real-time cancer diagnosis has been attempted during breast surgery [13]. Raman spectroscopy offers many opportunities for the development of sensitive diagnostic tools for rapid identification of

pathogenic microorganisms [86], assessing tissue specimens including tumor grade types [87, 88], performing continuous patient monitoring (e.g., blood analysis), guiding of surgical interventions, and intraoperative tumor border determination. Since the differentiation of tissue is possible, it is predicted that Raman spectroscopy may soon become a valuable tool to assist in clinical pathology [89].

Researchers at Wayne State University have been investigating the potential of Raman spectroscopy for tumor detection [56, 90, 91]. In 2005–6, 143 human tissues samples (22 from normal tissues and 121 from tumors) were studied and approximately 1700 Raman spectra were collected. This library of Raman data has been analyzed using a variety of statistical techniques, such as principal component analysis and discriminant function analysis. In addition, learning algorithms based on artificial neural networks and support vector machines have been developed to classify the data. Chapter 3 contains more information.

4.3.2 Image-Guided Surgery

During complex operations, surgeons must maintain a precise sense of three-dimensional anatomical relationships [92]. In addition, they must use their judgment, experience, and pathological evaluation (biopsy) to determine resection boundaries. Image-guided surgery fuses medical imaging, computer visualization, and real-time tracking of medical tools to provide the surgeon with a more detailed view of the patient's anatomy and the operating environment. In addition, image-guided surgery has allowed the development of minimally invasive surgical systems, which can greatly reduce cost, surgeon strain, and patient recovery time.

There are two different types of visualization technology that are especially applicable for the medical domain: augmented reality (AR) and virtual reality (VR) [93]. Traditional image guidance is an example of VR. Surgeons can view a computer-generated 3D scene showing their

tracked surgical tools with respect to a lesion inside a patient. This technology is now starting to be used in several branches of surgery, such as neurosurgery, spinal surgeries [94], orthopedic surgery [95], dental surgery, and even some examples of general surgery [96].

An AR system generates a composite view for the user that includes a live camera view fused (via registration) with computer-generated data (e.g., 3D geometry) or other sensed data [97]. AR is an extension of VR that represents a middle ground between computer graphics in a completely synthetically generated world (as in VR) and the real world [98-100]. In 2005, a system was developed at Wayne State University that simultaneously allows the surgeon to have both an AR and VR view of the patient's data [101].

Merging visualization technology with sensor technology will enable surgeons to accurately locate and classify tissues within the body. For example, Raman spectroscopy is a powerful sensing technology that can benefit from integration with advanced visualization techniques. Technology that integrates imaging and sensor information in real-time will add new dimensions to what can be done to diagnose and treat patients [102, 103].

4.4 Materials and Methods

The goal of this work is to create a system to detect cancerous tissue in the human body. However, since this chapter focuses on the integration and visualization of Raman spectroscopy with image-guided surgery, a phantom model can be used instead of actual tissue. Once the work is more developed, it can be tested on animal and human models. It is known from other literature and the research at Wayne State University that Raman spectroscopy is capable of detecting cancerous tissue quickly and accurately [13, 25, 40, 56, 91]. Therefore, this work has the potential to extend to real tissue.

Thus, in order to evaluate the integration of Raman spectroscopy and image-guided surgery, a system utilizing several components was developed (Figure 4.1). A portable Raman spectrometer was attached to a passively articulated mechanical arm. Classification algorithms for Raman spectra were also implemented. The results of the classification are sent to a medical visualization system. Once these systems were integrated together, testing was done with a phantom skull (shown in Figure 4.2). The skull was filled with various plastic and rubber objects, and CT images were obtained. The entire system was then used to scan objects in the skull, classify the resulting spectral data, and then place markers within the visualization system. Each of the subsystems is described in greater detail below.

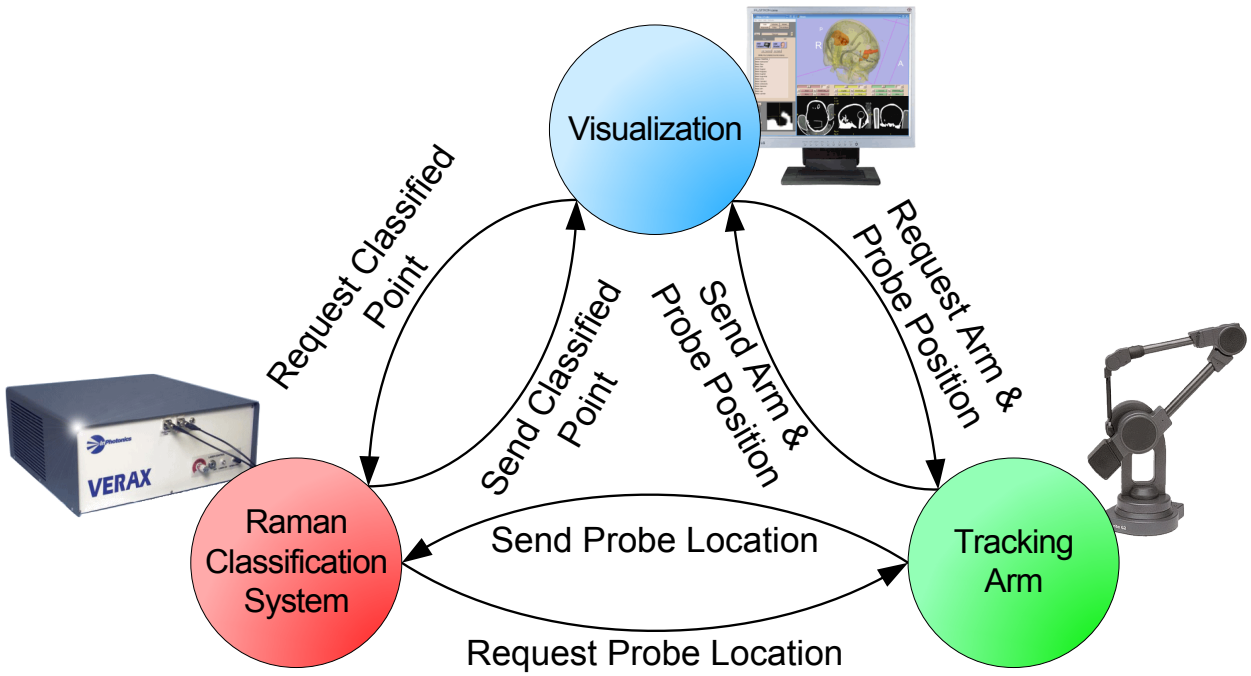

Figure 4.1: Organization of the system components and the software messages sent among them.

4.4.1 Tracking Arm

To track the position of a Raman spectrometer, one was attached to a passively articulated arm, a MicroScribe G2X (Immersion, San Jose, CA), as shown in Figure 4.2. This arm has five

degrees of freedom and, based on previous research [93], can track the location of its end-effector with an error less than 1 mm. It was chosen because it is simple to use and its tracking accuracy is within acceptable limits.

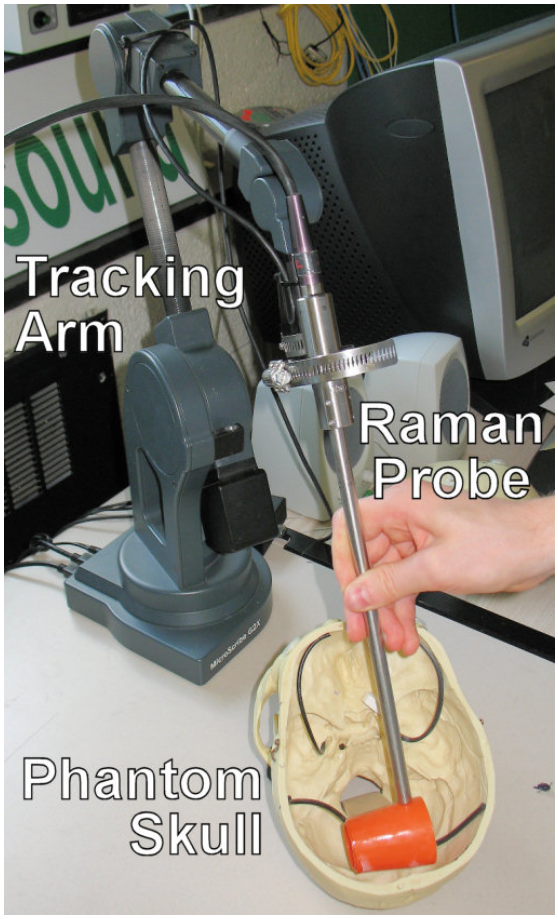

Figure 4.2: The Raman probe, attached to the end of the tracking arm, is used to scan a plastic cup and other objects within the phantom skull.

A software application was developed that registers the MicroScribe with patient imaging data and tracks the location of its end-effector. The tracking is accomplished by passing the arm's angular joint feedback through a mathematical model of the MicroScribe (a process known as forward kinematics). The tracking data is relayed in real-time to the visualization system. In the following section, the technical details of the tracking software are presented.

4.4.2 Medical Robot

The AESOP 1000 is a medical robot that was designed to position an endoscope during minimally invasive surgery. As described in [104], a custom control system was developed to enable computerized control of the robot. Further details on the robot can be found in section 5.6.2.1.

The tracking and control software for the robot was designed to integrate with the visualization systems of the image-guided surgery system [104]. Hence, it should be possible to use the robot to position and track a Raman probe. For this preliminary work, the robot held an endoscope instead of a Raman probe.

A robot would enable automated scanning of a region with a probe. The time required to scan a region would depend on at least three factors. The first is the movement speed of the robot, which may have to be lowered for safe use inside a patient. The second is the acquisition time of the Raman spectrometer. The Raman system used in this chapter requires 5–10 seconds to obtain a spectrum with a sufficient signal-to-noise ratio. The third factor is the processing time for the Raman data. This can be completed in well under a second, which means it is not a significant factor.

Depending on the number of points to be scanned and their locations, it could take several minutes to complete a series of scans with a robot-controlled probe. For a surgical application with long wait times for histopathology, this may be reasonable.

4.4.3 Tracking Algorithms

The model of the MicroScribe was expressed using Craig's modified Denavit–Hartenberg (DH) convention [28]. As described under section 2.3, the DH convention provides a compact

way to represent the structure of a robot (including link lengths, joint types, and joint angles). The DH parameters of the MicroScribe are shown in Table 4.1.

i	a_{i-1} (mm)	d_i (mm)	α_{i-1} (rad)	θ_i (rad)
1	0	210.82	0	(θ_1)
2	24.18	–22.53	0.4999π	(θ_2)
3	260.68	–0.30	-0.0002π	(θ_3)
4	13.89	234.70	0.4972π	(θ_4)
5	–10.26	8.10	-0.5007π	(θ_5)
6	10.16	–134.16	-0.4994π	0

Table 4.1: Denavit–Hartenberg parameters for the MicroScribe G2X mechanical arm. The non-zero (θ_i) terms represent variable joint angles.

Homogeneous transformation matrices are efficient mathematical structures that can be used to translate between different frames of reference [28]. The tracking system uses these matrices to establish the spatial relationships among the MicroScribe arm, Raman probe, patient (phantom skull), and patient imaging (CT scan data). This enables the software to track the location of the MicroScribe with respect to the phantom skull and overlay this information on the CT scan data in the visualization system.

In order to track the MicroScribe's end-effector relative to the skull, a registration must be performed between the two. To accomplish this, a pair-point-matching algorithm is executed using seven common fiducial points on the phantom skull and its CT scan data. The algorithm uses an iterative Levenberg–Marquardt optimization method to establish a homogeneous transformation from the skull (S) to the base of the MicroScribe (B), ${}^{S}\boldsymbol{T}_{B}$ [78, 105].

The next step in tracking the MicroScribe is to determine the location of its end-effector (EE) relative to its base. This is achieved via forward kinematics (described in section 2.3.1.1). Specifically, a series of transformation matrices are multiplied as shown in Equation 4.1. The first matrix (${}^{B}\boldsymbol{T}_{1}$) represents the change in position and orientation from the base coordinate

frame to the frame at joint 1, the second matrix (${}^{1}\boldsymbol{T}_{2}$) represents the change in position/orientation from the frame at joint 1 to the frame at joint 2, etc. Each transformation matrix is generated via Equation 2.10, which uses the joint angles (θ_i) and other DH parameters given in the rows of Table 4.1. Note that the final transformation to the end-effector corresponds to row 6.

$$ {}^{\mathrm{B}}\boldsymbol{T}_{\mathrm{EE}} = {}^{\mathrm{B}}\boldsymbol{T}_{1}\,{}^{1}\boldsymbol{T}_{2}\,{}^{2}\boldsymbol{T}_{3}\,{}^{3}\boldsymbol{T}_{4}\,{}^{4}\boldsymbol{T}_{5}\,{}^{5}\boldsymbol{T}_{\mathrm{EE}} \qquad \text{Equation 4.1} $$

Finally, the transformation from the skull to the base (${}^{\mathrm{S}}\boldsymbol{T}_{\mathrm{B}}$) can be combined with the transformation from the base to the end-effector (${}^{\mathrm{B}}\boldsymbol{T}_{\mathrm{EE}}$) to produce the desired transformation from the skull to the end-effector (${}^{\mathrm{S}}\boldsymbol{T}_{\mathrm{EE}}$). This is accomplished with Equation 4.2. The transformation from the skull to the end-effector allows the visualization system to display the location of the MicroScribe's tip relative to the 3D CT imaging of the phantom skull.

$$ {}^{\mathrm{S}}\boldsymbol{T}_{\mathrm{EE}} = {}^{\mathrm{S}}\boldsymbol{T}_{\mathrm{B}}\,{}^{\mathrm{B}}\boldsymbol{T}_{\mathrm{EE}} \qquad \text{Equation 4.2} $$

To track the Raman probe, which is attached to the MicroScribe, the model for the MicroScribe was extended by adding an extra transformation from the end-effector to the tip of the Raman probe (RP). This transformation is shown in Equation 4.3. The negative terms represent the offsets (in mm) of the probe's tip along the *x*-axis and *z*-axis of the end-effector's coordinate frame.

$$ {}^{\mathrm{EE}}\boldsymbol{T}_{\mathrm{RP}} = \begin{bmatrix} 1 & 0 & 0 & -14.2875 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -268.2875 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{Equation 4.3} $$

Combining the results of Equation 4.2 and Equation 4.3 allows the computation of the transformation from the skull to the tip of Raman probe (${}^{S}\boldsymbol{T}_{RP}$). This is done in Equation 4.4. The transformation allows the probe to be tracked in the visualization system relative to the skull's CT scan data.

$$ {}^{S}\boldsymbol{T}_{RP} = {}^{S}\boldsymbol{T}_{EE}\,{}^{EE}\boldsymbol{T}_{RP} \qquad \text{Equation 4.4} $$

4.4.4 Raman Spectrometer

The portable Raman spectrometer used in this research is a VERAX (InPhotonics, Norwood, MA). It is shown in Figure 4.3. This device uses a fiber optic probe and a 785 nm, 300 mW laser to obtain Raman spectra with an accuracy up to 4 cm^{-1}. The end of the probe is a long tube with a diameter of 1 cm, which is suitable for a test platform for minimally invasive surgery. In addition, the probe can be sterilized via an autoclave.

The Raman probe (shown in Figure 4.2) was affixed to the MicroScribe using a simple clamping system. The end-effector of the MicroScribe was marked to ensure consistent placement, allowing the probe to be detached and reattached.

Figure 4.3: The VERAX portable Raman spectrometer with an attached fiber optic probe.

4.4.5 Raman Classification

Many techniques have been developed for the classification of Raman spectra. For this implementation, a method based on artificial neural networks was used, which have been shown to perform well for Raman classification [22, 106, 107].

A variety of preprocessing tasks are performed on the raw Raman spectral data, including background fluorescence subtraction (via the "adaptive minmax" method [69]), median noise filtering, and normalization. Due to the high dimensionality of Raman spectra, principal component analysis [49] is used to capture the portions of spectral data with the highest variance and express them in a compact form (called scores). These scores are fed into the neural network.

The implemented neural network is a two-layer feedforward perceptron network. There is one input for each score produced during the preprocessing phase and one output for each possible class (e.g., plastic vs. rubber or healthy vs. cancerous tissue). The hidden layer uses hyperbolic tangent activation functions, and the output layer has logistic sigmoid activation functions. For this application, the network used 10 hidden neurons and was trained via backpropagation. The final output was the classification of the scanned tissue/material and a percentage indicating the confidence of the neural network.

As Raman scans are initiated with the image guidance system, the results of processing and classifying the spectra can be viewed on a monitor. An example of the output is shown in Figure 4.4.

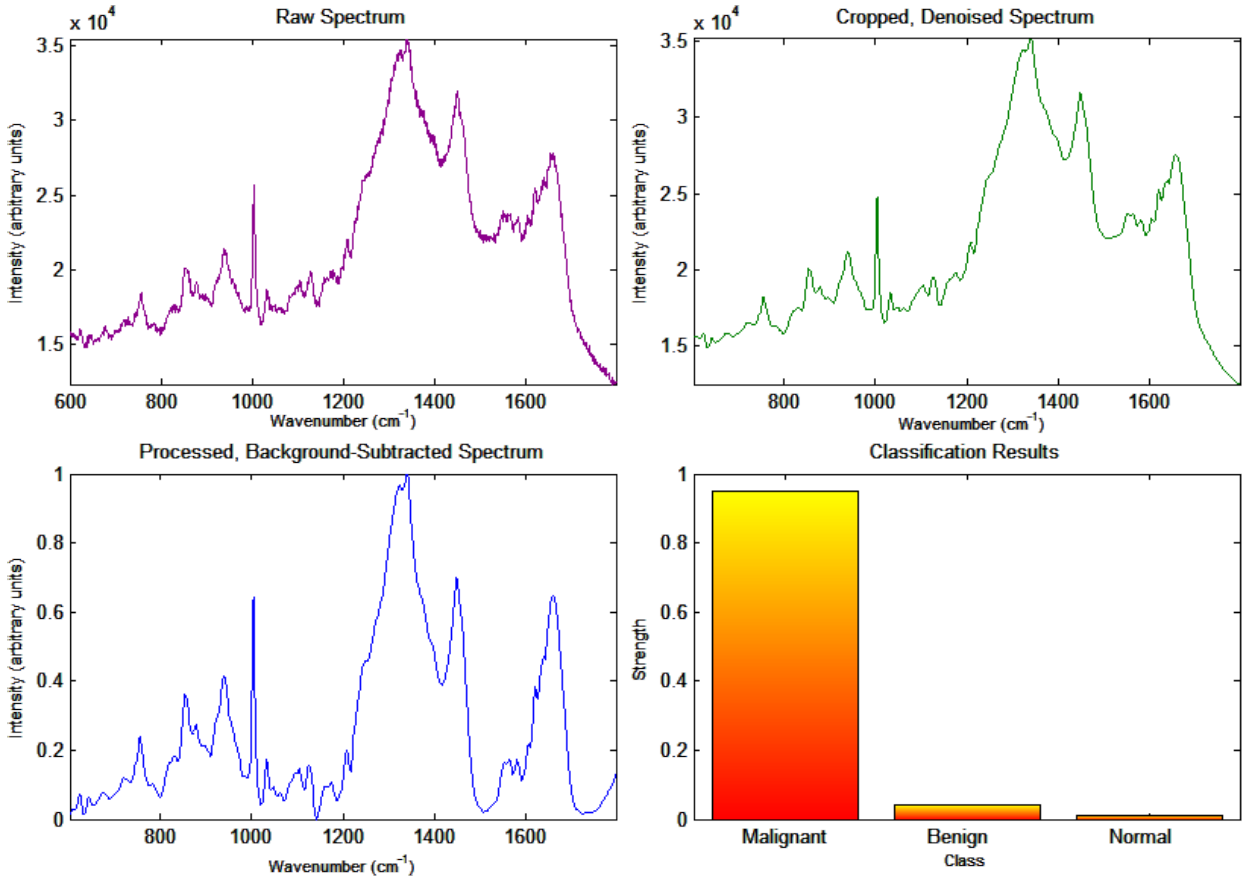

Figure 4.4: Image of the real-time classification application as it processes and classifies a Raman spectrum.

4.4.6 Virtual Reality Visualization

The visualization for the presented image-guided surgery system is implemented using 3D Slicer (www.slicer.org), an open-source application for displaying medical data. 3D Slicer provides a virtual reality environment in which various imaging modalities (e.g., CT or MRI data) can be presented. The software includes the ability to display the locations of objects with respect to 3D models that are derived from segmentation of the medical imaging.

3D Slicer was modified in several ways to adapt it for this application. First, a TCP/IP interface was developed that receives the tracking data for the MicroScribe and displays its position in the VR environment relative to the medical imaging data. This allows the Raman probe to be tracked in real-time. Second, a method was developed to place colored markers that

indicate tissue/material classification on the medical imaging data. The combination of these modifications enables the system to denote the location and classification of tissue/material scanned with the probe in near real time.

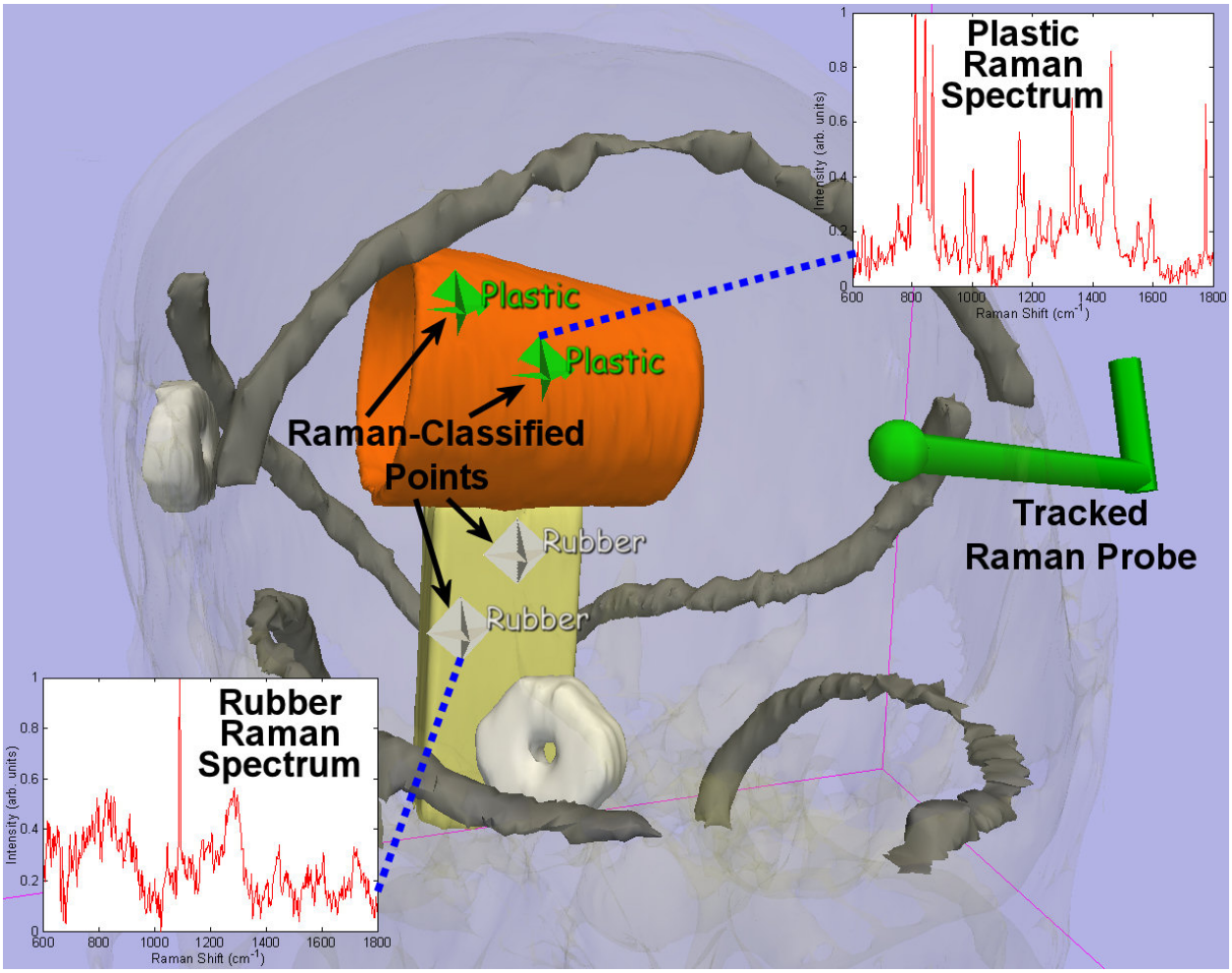

Figure 4.5: A screenshot of the visualization system showing 3D models (derived from CT imaging of the phantom skull), the location of the tracked Raman probe, collected Raman spectra, and colored markers denoting the classifications (computed from Raman scan data) of various objects within the skull.

4.4.7 Augmented Reality Visualization

An AR visualization module was also developed for the image guidance system. It uses a standard USB camera to acquire live images of the target object (e.g., the phantom skull). An optical tracking device, the Polaris Accedo (Northern Digital Inc., Ontario, Canada), was used to

track the position of the camera. To accomplish the tracking, spheres that reflect infrared light were affixed to the camera. This is shown in Figure 4.5.

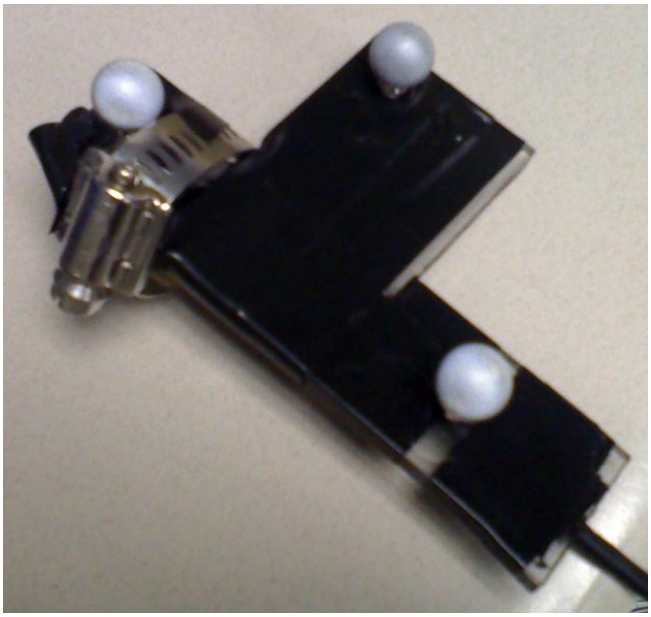

Figure 4.6: Standard USB camera with spherical markers affixed for optical tracking.

The optical tracker returns the transformation from the camera to the tracker. To obtain the camera's position with respect to the skull, a transformation was needed from the tracker's coordinate system to the skull. This was accomplished by registering the camera to the skull in the same manner as described in section 4.4.3. The tip of the camera was used to touch the fiducial points on the skull, enabling the required transformation to be computed.

With the position of the camera known with respect to the skull, the 3D models from the VR visualization can be placed on top of the video from the camera. In addition, the results of Raman analyses (classification markers) can also be displayed on the video. Figure 4.7 shows an image of the AR module in use.

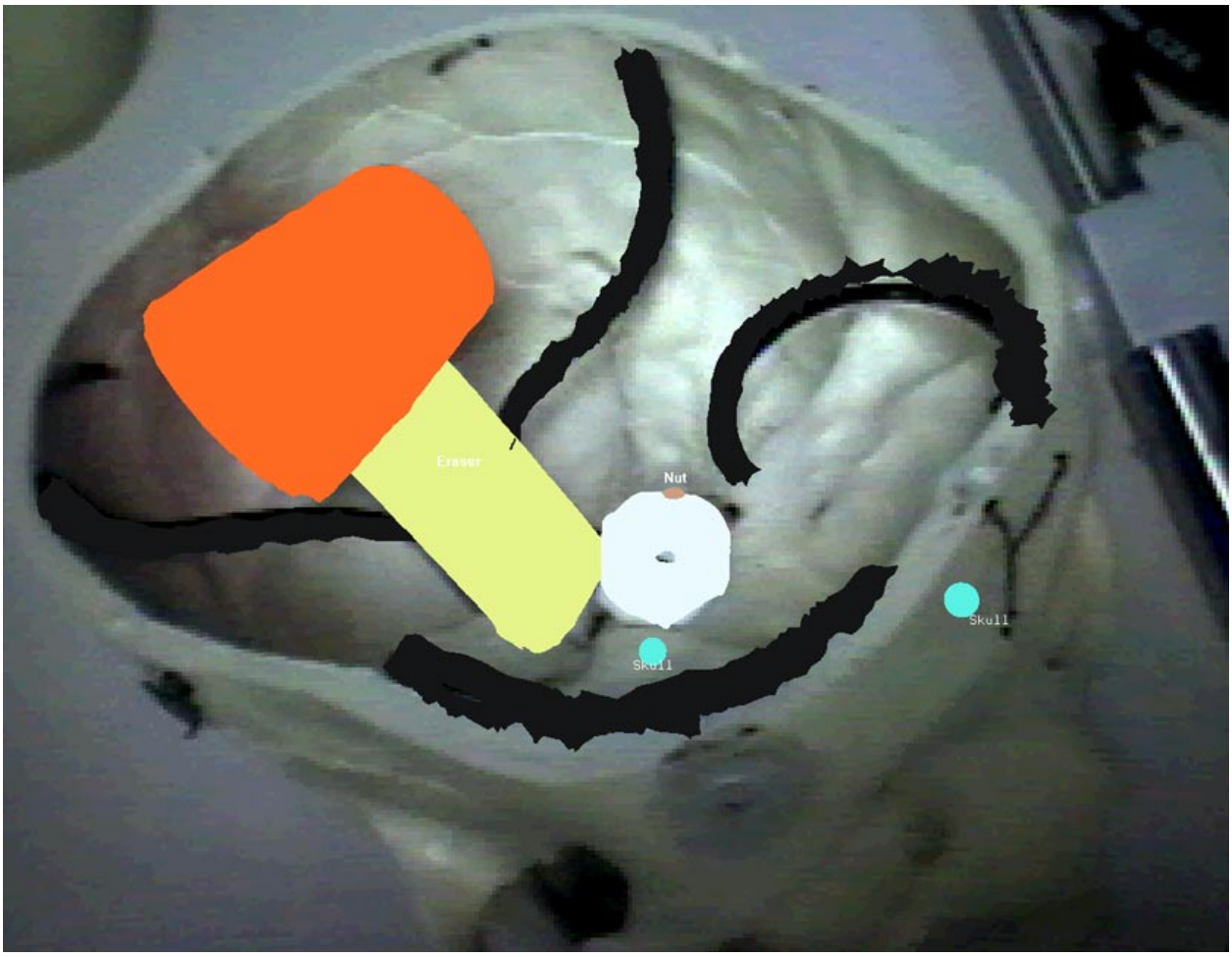

Figure 4.7: Augmented reality visualization module of the image guidance system. The 3D models are shown on top of the real objects in the skull, and the colored markers denoting Raman classifications can be seen.

4.5 Results

As described in section 4.4, the completed system was used to scan objects within a phantom skull. The MicroScribe and probe were positioned manually and tracked in real-time during this test. The collected Raman spectra were classified and displayed as colored markers in the visualization system. This is shown in Figure 4.5.

The system performed as expected. The tracking of the probe, the classification of the Raman spectra, and the display of the colored markers all occurred in real-time. The only major

delay was caused by the scanning of the tissue by the Raman probe, which requires roughly 5 seconds to produce a scan with a reasonable signal-to-noise ratio.

The Raman scans were able to distinguish between the plastic and the rubber objects. The corresponding markers in the visualization display correctly reflected the classifications that were made. The positions of the markers were also accurate with respect to the locations from which scans were taken. Since the setup of the system is very similar to that of some previous work [93], it is estimated that the probe tracking accuracy is around 1 mm.

4.5.1 Preliminary AESOP 1000 Results

The AESOP 1000 medical robot was also tested with the VR module of the system. Unfortunately, errors in its DH model made it insufficient for accurate sensor placement. The tracking system and measured errors of the AESOP 1000 are described in [104]. These errors were the motivation to create the kinematic calibration technique of Chapter 5.

4.6 Discussion and Future Work

In a paper discussing the future of robotic surgery [108], Taylor and Stoianovici stated that it will become increasingly important to design systems that can incorporate a wide range of biomedical sensors and that can work with multiple imaging modalities. An example of this goal has been provided via the successful integration of Raman spectroscopy with an image-guided surgery system, demonstrated using a phantom skull. The presented system, which tracks a registered Raman probe, is developed in a manner that allows the results of neural network-classified Raman data to be displayed directly on any imaging modality of the phantom's anatomy, all in near real-time.

This chapter demonstrates that Raman spectroscopy and image-guided surgery can be combined to provide a powerful diagnostic system. Even though a phantom model was used, the

underlying technologies have been previously shown to work with human tissue. With further research and development, this system should be suitable for human applications. For now, the system will continue to be developed and tested using phantom models. In the future, the plan is to evaluate the system with animal testing. Eventually, the application of the work to human cases is anticipated. If the results continue to be positive, it will demonstrate that Raman spectroscopy has the potential to be a powerful complement to conventional histopathology.

To the author's knowledge, there had been no prior prototypes in the literature that attempted to combine Raman spectroscopy and image-guided surgery before this work was published [20]. It is hypothesized that a system based on these technologies could eventually provide many benefits in the surgical environment. These benefits could include faster diagnoses and more accurate resections, hence producing better patient outcomes. However, there are certain issues that need detailed research and development.

One key issue is the size of the Raman probe. Ideally, the entire instrument should be about the size of a scalpel for easy manipulation by the surgeon. Existing portable systems are still too bulky as real-time diagnostic tools for certain applications. The availability of miniaturized hand-held devices would create a platform for a wider range of research on real-time tissue diagnostics. Once developed, this technology will enable the creation of a whole new field of real-time, *in vivo* spectroscopic diagnostics of tissue.

Another relevant issue is the positional accuracy of the image-guided Raman system. Although the tip of the employed Raman probe can be tracked to within an acceptable accuracy for neurosurgery, there is an uncertainty of about 5 mm (the diameter of the probe's transmission window) as to exactly where the human-held probe actually measures the point of interest. A system in which the actual point of interest could be better determined and more accurately

placed would be much more useful. The issues of placement accuracy and the requirement of holding the sensor still during acquisition can be alleviated by the closely related field of medical robotics. A robot can accurately place and steadily hold a probe in the surgical environment.

Thus, the plan is to expand the work described in this chapter to include the AESOP medical robot (Intuitive Surgical, Sunnyvale, CA). This robot will be used to actively position the Raman probe instead of the passive MicroScribe arm. This will provide several benefits over the current system. First, the AESOP will be able to hold the probe completely steady during a scan, which is important considering scans can take 5 or more seconds. Second, the AESOP will be able to position the probe more accurately using medical imaging data, enabling the surgeon to precisely scan the tissue of interest. Third, automated scanning will be possible, in which the AESOP moves among numerous points over a defined area of tissue, taking a Raman scan at each.

The work with the AESOP will include a detailed human (surgeon) factors study. Tests will be performed to ensure that the system is beneficial to surgeons. For example, it must be verified that the speed and accuracy of tissue diagnoses are improved through the use of the system. Moreover, several visualization techniques will be implemented and compared to find the best one. Such testing is necessary during research and development to ensure that the results are a useful addition to the field.

Chapter 5:
Fixed-Point Kinematic Calibration

5.1 Introduction

When positioning a sensor, it is important for the robotic manipulator to have sufficient accuracy. If the sensor is placed at the wrong location, the acquired data may be insufficient or incorrect. A number of issues can contribute to error in the robot's positioning, such as manufacturing tolerances, imprecise assembly, and component deformation. A process known as kinematic calibration can be executed to reduce the level of error.

This chapter describes a calibration technique that can improve the positional accuracy of a serial manipulator. It consists of easily performed procedures and a set of computer algorithms. It requires no external measurement system, and it overcomes some of the limitations of existing techniques. It can be used for the improvement of sensor placement or any other application that involves positioning or tracking based on a kinematic model.

5.2 Overview

The kinematic calibration technique presented in this chapter is an extension of earlier work in closed-loop calibration (section 5.3). It involves placing the end-effector of a serial manipulator at a fixed point and then altering the orientation of the end-effector while maintaining contact with the point. Using only the internal joint sensors of the robot, it is possible to collect enough information to calibrate most of the parameters of the robot's model.

The calibration technique consists of four phases (section 5.4). First, joint readings of the robot are acquired when the robot touches a fixed point from multiple orientations. Second, the collected data is processed to generate a calibrated model. Third, the joint values of the robot are

recorded when the robot touches specific points on a known object. Finally, this data is used to correct certain deficiencies in the calibrated model and enable the robot to operate in an application-specific coordinate system.

The presented calibration technique was evaluated in three experiments (section 5.6). The first two experiments were performed in a simulated environment using four different robot models. One simulation was ideal, and the other added noise to the joint feedback. The final experiment tested the calibration algorithms on two actual manipulators with real measured data.

5.3 Background and Related Work

5.3.1 Kinematic Calibration

Typically, a robot's kinematic model is provided by the manufacturer or can be developed from its specifications. However, due to errors/tolerances in the construction or possibly the wear of normal usage, the model will not match the physical structure of the robot. Kinematic calibration is a process that attempts to minimize the differences between the kinematic model of a robot and the actual structure of the robot.

It is generally not feasible to attempt a direct measurement of the robot's parameters (e.g., link lengths and angles) using simple tools like a ruler. The accuracy would be limited, and the physical structure of the robot often makes it difficult or impossible to appropriately place a measuring tool. Therefore, kinematic calibration often relies on measurements of the end-effector's position and orientation (pose) or some subset of this information. It then uses mathematics to indirectly identify a more accurate version of the robot's parameters.

The two most commonly used types of kinematic calibration are open-loop methods and closed-loop methods [109]. Open-loop methods require an external measurement system (e.g., a laser interferometer or coordinate measuring machine) to assess the pose of the end-effector

[105]. This adds to the cost and complexity of the process, and the measurement device can introduce its own error. Closed-loop methods, on the other hand, only require the internal joint feedback of the robot to perform the calibration [110, 111]. The method described in this chapter (fixed-point calibration) can be classified as closed loop.

5.3.2 Open-Loop Kinematic Calibration

The traditional form of calibration is open-loop calibration. It requires the position and/or orientation of the end-effector to be determined via an external measurement system. It is called "open loop" because the end-effector is positioned freely in space (instead of being constrained by contact with the environment).

This section provides an overview of the fundamental concepts used in open-loop calibration. Many of these concepts also apply to closed-loop (and fixed-point) calibration. For more detailed information, there are numerous references that explain the mathematical background of open-loop kinematic calibration [105, 111]. Section 2.3 also provides some relevant background information on robot modeling.

The goal of kinematic calibration is to optimize the parameters of a robot's kinematic model so that calculations using the model match the performance of the actual robot as closely as possible. For example, a calibrated model can be used to more accurately track and move the robot. In order to perform calibration, it is assumed that an initial model of the robot is available.

In open-loop calibration, parameters of the model are adjusted so that the pose of the robot calculated via the model (and internal joint measurements) matches the pose determined via external measurements. Thus, a number of measurements are needed to perform open-loop calibration. The measurements include the joint feedback from the sensors built into most robots

(e.g., encoders) and the output of an external measurement system (e.g., a coordinate measuring machine).

Typically, the first step in acquiring a set of measurements is to place the robot into a particular pose. Next, the current set of variable joint values (θ_i^{var} or d_i^{var}) is recorded using the robot's controller and its internal sensors. Note that a set of joint values that corresponds to a specific pose of the robot will be referred to as a *joint configuration*. After that, the external measurement system is used to record the pose of the robot (usually of its end-effector) or some subset of its pose.

Due to the number of unknown parameters and sources of noise/error, multiple sets of measurements must be taken. Therefore, the joint values and position/orientation of the robot are measured when the robot is placed into many different poses/joint configurations. Different techniques are available for choosing sets of poses that improve the observability of the parameters [109, 111].

5.3.2.1 Mathematical Formulation of Open-Loop Parameter Identification

Open-loop calibration compares the calculated pose of the robot to its measured pose. The pose of a robot (especially of its end-effector) is computed using forward kinematics (section 2.3.1.2). These calculations require a complete set of DH parameters for the robot. For the purpose of calibration, the DH parameters are divided into two groups.

The first group, $\boldsymbol{c}_q$, represents the variable joint values (θ_i^{var} or d_i^{var}) that are associated with a particular pose (q) of the robot. These values define the joint configurations, which are obtained via measurement of the robot's internal sensors in the manner described above. Although techniques are sometimes used to calibrate the sensors, the measured joint values are not modified during a kinematic calibration.

The second group of DH parameters, $\boldsymbol{m}$, contains the non-variable ones; they do not change when the pose of the robot changes. These parameters described the fixed structure of the robot, including the fixed offsets (θ_i^{off} or d_i^{off}) for each variable joint. Therefore, they are the ones that need to be modified to produce a calibrated robot model.

The two groups of DH parameters ($\boldsymbol{c}_q, \boldsymbol{m}$) are both used to compute the forward kinematics of the robot. The output is the pose (position and orientation), $\boldsymbol{y}_q$, of the robot's end-effector. This process is expressed in Equation 5.1.

$$\boldsymbol{y}_q = \boldsymbol{f}(\boldsymbol{c}_q, \boldsymbol{m}) = \boldsymbol{f}_q(\boldsymbol{m}) \qquad \text{Equation 5.1}$$

The following terms are used in Equation 5.1 or are necessary to define those terms:

- n_{pc} is the number of pose components being measured and calculated in the calibration. Six components can be used to completely specify the position and orientation of the robot's end-effector.
- $\boldsymbol{y}_q$ is an $n_{\text{pc}}\times 1$ vector that contains the calculated pose of the end-effector for joint configuration q. For a complete pose, $\boldsymbol{y}_q$ often has the form $[p_x, p_y, p_z, \varphi, \theta, \psi]^{\text{T}}$. This vector contains the position (p_x, p_y, p_z) and orientation (φ, θ, ψ) of the end-effector expressed using Cartesian coordinates and Euler angles (per section 2.3.1.1). If any pose components are not being used (such as the orientation), the corresponding variables are omitted from $\boldsymbol{y}_q$.
- $\boldsymbol{f}$ is a function that computes the forward kinematics of the robot using the variable and non-variable DH parameters. Section 2.3.1.2 has a detailed description.
- n_{j} is the number of joints in the robot.

- $\boldsymbol{c}_q$ is a vector containing the measured joint values (θ_i^{var} or d_i^{var}) for joint configuration q. Its length is equal to n_{j}.
- $\boldsymbol{m}$ is $4n_{\text{j}}\times 1$ vector that contains all the non-variable DH parameters of the robot's model. It holds the sets of four fixed DH parameters for all joints. For example, in the original form of the DH convention, it would contain $(\alpha_i, a_i, \theta_i^{\text{off}}, d_i)$ for revolute joints or $(\alpha_i, a_i, \theta_i, d_i^{\text{off}})$ for prismatic joints.
- $\boldsymbol{f}_q$ is the same as $\boldsymbol{f}$ with $\boldsymbol{c}_q$ as an implicit input. This is used to simplify the notation since the variable joint values, $\boldsymbol{c}_q$, are not modified during kinematic calibration.

Considering the description given above, the objective of open-loop calibration is to modify the non-variable parameters, $\boldsymbol{m}$, so that the difference between the calculated pose, $\boldsymbol{y}_q$, and the measured pose is minimized. To accomplish this goal, a more direct relationship between changes in the pose and changes in the parameters is useful. This relationship can be found by through a Taylor series expansion of $\boldsymbol{f}_q(\boldsymbol{m})$ at a particular set of parameters, $\boldsymbol{m}_k$. This is shown in Equation 5.2 with the higher order terms omitted (making the relationship an approximation).

$$\boldsymbol{f}_q(\boldsymbol{m}) = \boldsymbol{f}_q(\boldsymbol{m}_k) + \left.\frac{\partial \boldsymbol{f}_q(\boldsymbol{m})}{\partial \boldsymbol{m}}\right|_{\boldsymbol{m}=\boldsymbol{m}_k} (\boldsymbol{m} - \boldsymbol{m}_k) + \cdots$$
$$\boldsymbol{f}_q(\boldsymbol{m}) \approx \boldsymbol{f}_q(\boldsymbol{m}_k) + \boldsymbol{J}_{q,k}(\boldsymbol{m} - \boldsymbol{m}_k) \qquad \text{Equation 5.2}$$

In Equation 5.2, $\boldsymbol{J}_{q,k}$ is a $n_{\text{pc}}\times 4n_{\text{j}}$ Jacobian matrix that provides the partial derivatives of the functions for each pose component of $\boldsymbol{y}_q$ with respect to each of the model parameters in $\boldsymbol{m}$. These functions and partial derivatives can be derived from the forward kinematics equations, but the process is lengthy and not needed here. The reader can find the derivations in other literature [105, 111].

Equation 5.3 shows the form of the Jacobian matrix $\boldsymbol{J}_{q,k}$. The functions denoted by $f_{\text{component},q}$ are equivalent to $\boldsymbol{f}_q$ but with only the specified pose component being calculated. One point of interest is that the partial derivatives evaluate to single numbers when a specific joint configuration ($\boldsymbol{c}_q$) and a specific set of model parameters ($\boldsymbol{m}_k$) is used.

$$\boldsymbol{J}_{q,k} = \left.\frac{\partial \boldsymbol{f}_q(\boldsymbol{m})}{\partial \boldsymbol{m}}\right|_{\boldsymbol{m}=\boldsymbol{m}_k} = \left[\begin{matrix} \frac{\partial f_{p_x,q}(\boldsymbol{m})}{\partial \alpha_1} & \frac{\partial f_{p_x,q}(\boldsymbol{m})}{\partial a_1} & \cdots & \frac{\partial f_{p_x,q}(\boldsymbol{m})}{\partial d_{n_j}} \\ \frac{\partial f_{p_y,q}(\boldsymbol{m})}{\partial \alpha_1} & \frac{\partial f_{p_y,q}(\boldsymbol{m})}{\partial a_1} & \cdots & \frac{\partial f_{p_y,q}(\boldsymbol{m})}{\partial d_{n_j}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_{\psi,q}(\boldsymbol{m})}{\partial \alpha_1} & \frac{\partial f_{\psi,q}(\boldsymbol{m})}{\partial a_1} & \cdots & \frac{\partial f_{\psi,q}(\boldsymbol{m})}{\partial d_{n_j}} \end{matrix}\right]_{\boldsymbol{m}=\boldsymbol{m}_k} \quad \text{Equation 5.3}$$

If an iterative process is used to find the optimal parameters, the next set of parameters in the sequence will be $\boldsymbol{m}_{k+1}$. This can be put into Equation 5.2 in place of $\boldsymbol{m}$ to clarify the iterative procedure. This is shown in Equation 5.4 along with some consolidation of terms.

$$\begin{gathered} \boldsymbol{f}_q(\boldsymbol{m}_{k+1}) \approx \boldsymbol{f}_q(\boldsymbol{m}_k) + \boldsymbol{J}_{q,k}(\boldsymbol{m}_{k+1} - \boldsymbol{m}_k) \\ \boldsymbol{f}_q(\boldsymbol{m}_{k+1}) - \boldsymbol{f}_q(\boldsymbol{m}_k) \approx \boldsymbol{J}_{q,k}(\boldsymbol{m}_{k+1} - \boldsymbol{m}_k) \\ \Delta \boldsymbol{f}_q \approx \boldsymbol{J}_{q,k}\Delta \boldsymbol{m} \\ \Delta \boldsymbol{y}_q \approx \boldsymbol{J}_{q,k}\Delta \boldsymbol{m} \end{gathered} \quad \text{Equation 5.4}$$

In Equation 5.4, the following new terms are used:

- $\Delta \boldsymbol{y}_q$ and $\Delta \boldsymbol{f}_q$ represent the incremental change in pose that occurs for an incremental change in the non-variable DH parameters
- $\Delta \boldsymbol{m}$ represents an incremental change in the non-variable DH parameters

Equation 5.4 represents a linear model that approximates the relationship between the calculated pose and the parameters. This model can be used to help find the optimal set of parameters. In open-loop calibration, the error is defined by the difference between the

calculated pose, $\boldsymbol{y}_q$, and the measured pose. To correct the error, the incremental change in pose, $\Delta\boldsymbol{y}_q$, can be set according to the difference between the calculated and measured poses. Furthermore, the elements of $\boldsymbol{J}_{q,k}$ can be computed using the partial derivatives evaluated at the current set of joint values ($\boldsymbol{c}_q$) and the current DH parameters ($\boldsymbol{m}_k$). Consequently, all that remains is to solve for the incremental change in the parameters ($\Delta\boldsymbol{m}$) that will effect the corrections to the calculated pose.

In the linear model of Equation 5.4, n_{pc} equations can be written (six at most), but there are $4n_{\text{j}}$ unknown parameters, and manipulators have multiple joints. Therefore, it is not possible to solve for an optimized set of DH parameters using Equation 5.4 with the measured data from a single joint configuration, q. However, by using the measured data from multiple joint configurations, a sufficient number of equations can be formed to solve for the unknown parameters. Equation 5.5 represents an extended form of Equation 5.4 that incorporates the measured data from all the joint configurations into which the robot was placed.

$$\Delta\boldsymbol{y} \approx \boldsymbol{J}_k \Delta\boldsymbol{m} \qquad \text{Equation 5.5}$$

The following terms are used in Equation 5.5 or are necessary to define those terms:

- n_{jc} is the number of joint configurations (poses) from which measurements were obtained.
- $\Delta\boldsymbol{y}$ is the $n_{\text{pc}}n_{\text{jc}}\times 1$ vector that results from stacking the incremental changes in pose for each joint configuration: $\left[\Delta\boldsymbol{y}_1, \Delta\boldsymbol{y}_2, \ldots, \Delta\boldsymbol{y}_{n_{\text{jc}}}\right]^{\text{T}}$.
- $\boldsymbol{J}_k$ is the $n_{\text{pc}}n_{\text{jc}}\times 4n_{\text{j}}$ Jacobian matrix that results from stacking the individual Jacobian matrices for each joint configuration: $\left[\boldsymbol{J}_{1,k}, \boldsymbol{J}_{2,k}, \ldots, \boldsymbol{J}_{n_{\text{jc}},k}\right]^{\text{T}}$. It is still a function of the

current set of parameters, $\boldsymbol{m}_k$. $\boldsymbol{J}_k$ is sometimes referred to as the *identification Jacobian*.

Using the extended linear model of Equation 5.5, it is now possible to (attempt to) solve for $\Delta\boldsymbol{m}$. First, $\Delta\boldsymbol{y}$ and $\boldsymbol{J}_k$ are generated as described earlier for Equation 5.4. Due to noise and other sources of error, there is no exact solution for $\Delta\boldsymbol{m}$. Instead, the pseudoinverse of $\boldsymbol{J}_k$ can be used to find a best fit (least-squares) solution. This is shown in Equation 5.6.

$$\Delta\boldsymbol{m} \approx \left(\left(\boldsymbol{J}_k^{\mathrm{T}}\boldsymbol{J}_k\right)^{-1}\boldsymbol{J}_k^{\mathrm{T}}\right)\Delta\boldsymbol{y} \qquad \text{Equation 5.6}$$

Once the incremental change in the parameters has been found, the current set of parameters, $\boldsymbol{m}_k$, can be updated according to Equation 5.7. Since the linear model of the relationship between the calculated pose and the parameters is actually an approximation of a more complex nonlinear relationship, the first updated set of parameters is likely not the optimal solution. Instead, Equation 5.6 and Equation 5.7 should be iteratively applied until the incremental change is sufficiently small. This corresponds to the Gauss–Newton algorithm [105], which can work well if the initial set of parameters is reasonably accurate. Related (but slightly more complex) techniques like the Levenberg–Marquardt algorithm [105] can also be applied for additional robustness, especially if the initial set is not as accurate.

$$\boldsymbol{m}_{k+1} = \boldsymbol{m}_k + \Delta\boldsymbol{m} \qquad \text{Equation 5.7}$$

5.3.2.2 Identification Jacobian

The identification Jacobian ($\boldsymbol{J}_k$) from Equation 5.5 in the previous section is very useful. At a high level, it relates changes in each pose component to changes in each of the non-variable DH parameters. Consequently, it can be used to determine what changes are necessary in the model parameters to effect a desired change in the pose.

Analysis of the identification Jacobian can reveal lot of information about the robot's model with respect to the calibration. For example, if the identification Jacobian matrix has full rank, it indicates that all parameters of the model are identifiable. However, if the matrix is rank deficient, some parameters will not be (uniquely) identifiable.

A column of all zeros indicates that modifying the corresponding parameter has no effect on any of the measurable pose components. Consequently, that parameter cannot be identified. It is generally excluded from the calibration to avoid creating problems for the calibration algorithm.

If columns of the identification Jacobian matrix are linearly dependent, then the parameters corresponding to those columns are said to be linearly dependent. In this case, it is not possible to find the exact values of each of the parameters in the linearly dependent set. However, the value that represents the combined effect of the parameters can generally be found. Therefore, sometimes certain parameter values are excluded from calibration so the algorithm doesn't try to assign specific values to parameters that cannot be uniquely determined. Instead, just one parameter for a linearly dependent set will be calibrated. The identified value will not match the true value, but the overall effect of the linearly dependent parameters should be correct.

More rigorous approaches to analyzing the identification Jacobian are found in the literature [105, 109-112]. For example, by performing a singular value decomposition or QR decomposition of the matrix and analyzing the results, it is possible to classify parameters as identifiable, unidentifiable, weakly identifiable, or only identifiable in linear combinations. Once the parameters are better understood, they can be handled by the calibration algorithm in an appropriate fashion.

5.3.3 Closed-Loop Kinematic Calibration

Closed-loop kinematic calibration differs from open-loop calibration in an important way: it requires no external device to measure the pose of the robot. Instead, a closed-loop technique only requires the internal joint feedback of the robot. It is called "closed loop" because the end-effector is constrained by contact with the environment, forming a closed loop via the ground.

Closed-loop calibration is able to work without external measurements because of physical constraints on the end-effector. By using the environment to constrain some or all of the end-effector's pose (position and orientation) components, information about the end-effector's pose is known without acquiring any external measurements. Although the amount of known information about the pose is often less than in open-loop calibration, the data is generally sufficient to calibrate most of the kinematic parameters of a manipulator's model.

Closed-loop calibration is closely related to self-calibration and autonomous calibration. Sometimes, the terms are used synonymously. In other cases, "self-calibration" or "autonomous calibration" refers to calibration using additional sensors on the robot instead of external systems. They are often associated with parallel robots or the use of integrated vision systems. This chapter focuses on closed-loop calibration for serial manipulators using internal joint feedback and end-effector constraints.

This section provides an overview of some important concepts in closed-loop calibration. These concepts also apply to fixed-point calibration, which is a subset of closed-loop calibration. For more detailed information, the work by Bennett and Hollerbach [110] and other references [109, 111] can be consulted. Section 2.3 also provides some relevant background information on robot modeling.

There has been much research on the subject of closed-loop calibration. For example, work has been done where the robot's end-effector is constrained via a hinge joint [113], movement along a plane [114, 115], or movement along a cylinder [116]. It has been applied to parallel and cooperative robots/arms using multiple closed loops [117-120]. In addition, the technique has been used to calibrate both a robot and the parameters of a camera system [121]. Wampler et al. devised a method called the implicit loop method that unifies that mathematics of open-loop and closed-loop (including point-contact) calibration [122]. Their method incorporates *a priori* estimates of measurement and parameters errors, and it can be applied to parallel manipulators.

To perform closed-loop calibration, several requirements must be met. As with open-loop calibration, the ability to measure the variable joint values (joint configuration) of the robot is needed. In addition, the robot must have a sufficient number of degrees of freedom to permit movement in the robot's arm while the constraints on the end-effector are being enforced. Furthermore, a way to move the robot while respecting the end-effector constraints is required (e.g., manual positioning or a fixture and force control). It is also assumed that an initial model of the robot is available.

One weakness of open-loop calibration, which is stated in [110], is that it requires a length component of the robot to be precisely known in order to set the robot's scale. It can also be more difficult to place the robot into the multiple constrained poses needed by the method. However, the simplicity and savings afforded by closed-loop calibration can make it worthwhile.

Closed-loop calibration is also unable to determine the position and orientation of the robot's base (or equivalently, its end-effector) with respect to some external (world) coordinate system, as mentioned elsewhere [118, 123-125]. The robot is only calibrated with respect to the base of

its model, and even then there may be ambiguities due to unidentifiable parameters (per section 5.3.4.1). This problem is easier to solve in open-loop calibration with its external measurements.

5.3.3.1 Execution of Closed-Loop Calibration

Closed-loop calibration requires constraints to be imposed on the end-effector's pose, for which there is a variety of possibilities. For example, three different constraints were discussed in [110]. The complete pose (position and orientation) of the end-effector can be fixed, the complete position and two of three orientation components can be fixed (only allowing rotation like that of a hinge joint on a door), or just the position components can be fixed. In the last example, the end-effector is kept at a fixed point, but its orientation can vary freely. This is called the fixed-point or point-contact constraint.

Similar to open-loop calibration, multiple sets of measurements of the joint values must be recorded when the robot is placed into different poses. This ensures that a sufficient number of equations can be written to solve for the many unknowns of a robot model's kinematic parameters. In addition, multiple measurements must be taken under the same constraints. For example, with a fixed-point constraint, the same fixed point must reached (from various orientations) by multiple recorded poses/joint configurations. The consistency of the constraint across a number of measurements is what enables the constraint to substitute for external measurements.

There are at least two approaches to calibrating a robot model using closed-loop measurements. In the first, the constrained portion of the end-effector's pose is estimated using the current model of the robot and the measured joint configurations. For instance, with a fixed-point constraint, the position of the fixed point can be estimated by using the forward kinematics equations (section 2.3.1.2) to calculate the position of the end-effector for each joint

configuration and then finding the average of the results. Then an algorithm is used to modify the robot parameters until the constrained portion of the calculated poses converge upon the estimated constrained pose. This approach was mentioned by Craig [126], and it is the approach used in this chapter. Thus, it is explained in detail under section 5.4.4.

The second approach to calibrating a model with closed-loop data was described by Bennett and Hollerbach [110]. It applies an important modification to enable the techniques of open-loop calibration to be applied to closed-loop calibration [111]. Specifically, the constrained position and/or orientation of the end-effector are used to define the origin of a new base coordinate system. In other words, the true values of all constrained pose components of the end-effector are defined to be 0 in a new coordinate system. The definition of any components of the new coordinate system not related to constrained pose components is arbitrary.

For example, for a fixed-point constraint, the true position of the fixed point would be defined as (0, 0, 0) in the new base coordinate system. Since the orientation of the end-effector is not constrained, the orientation of the new coordinate system (with respect to the robot's original base coordinate system) is arbitrary.

To use this new coordinate system, the relationship between the robot's original base coordinate system and the new system must be specified. This can be defined by a transformation from the constrained position/orientation of the end-effector to the robot's original base coordinate system. Although there are many ways to represent a transformation (such as a homogeneous transformation matrix), it can be conveniently expressed as an additional set of four DH parameters (or two if four parameters is not enough). Thus, the transformation from the constrained end-effector pose back to the robot's original base is treated

like an additional link in the robot's kinematic chain. This transformation creates a closed loop in the robot model, so it will be referred to as the *loop-closing transformation*.

If the robot has n_j joints and its original base coordinate system has the subscript 0, then the new base coordinate system can be given the subscript −1. The coordinate system at −1 is partially or fully coincident with the coordinate system at the end-effector (subscript n_j) as needed to ensure the constrained pose components have a value of 0. Therefore, when using forward kinematics to calculate the pose of the end-effector with respect to coordinate system −1, the desired output is for all the constrained pose components to have a value of 0. Any deviation of these components from 0 is due to errors in the kinematic parameters ($\boldsymbol{m}$) or other sources of noise/error in the system.

With the loop-closing transformation in place, the desired values of the constrained end-effector components are known and defined to be 0. This is analogous to having measured the end-effector's pose with an external device and finding the values to be 0. Thus, the closed-loop calibration can be treated like an open-loop calibration and executed with the same mathematical techniques (section 5.3.2.1) with a few minor modifications.

To utilize an open-loop algorithm with closed-loop data, the definition of the error has to change (to the difference between the calculated constrained end-effector pose and all 0s). In addition, any unconstrained pose components have to be removed from consideration because their values are unknown. Furthermore, the DH parameters representing the loop-closing transformation have to be added to the robot's model ($\boldsymbol{m}$) and included in the calibration. They can be ignored when the calibration is complete.

5.3.4 Fixed-Point Kinematic Calibration

As mentioned in the previous section, fixed-point kinematic calibration is a specific type of closed-loop calibration. It requires the robot's end-effector have its position constrained to a fixed location while the orientation of the end-effector is altered. Using internal measurements of multiple joint configurations under this constraint, it is possible to calibrate the robot without using any external measurement devices.

One approach to fixed-point calibration was mentioned by Craig [126]. It seems to be similar to the approach used in this chapter (at least in terms of the error function), but insufficient detail was provided, and no results were presented. On the other hand, the approach suggested by Bennett and Hollerbach [110, 111] was described in much greater detail. In [110], they provide much of the mathematical background for open-loop calibration. They also include the results of several simulations.

The "point-contact" method of [110] corresponds to the fixed-point method described in this chapter. Hence, the point-contact method will be the focus of this section. Some of the limitations of that work will be highlighted, and comparisons will be made with the fixed-point method presented in this chapter.

Other authors have based their work on the point-contact method of [110]. Meggiolaro et al. used the technique to successfully calibrate a hydraulic robot with six degrees of freedom [127, 128]. Haidegger et al. applied the method to a neurosurgical robot [129]. Abtahi et al. extended the technique to the calibration of a parallel manipulator with six legs [130]. Gatla et al. extended the point-contact method so that it does not require the robot to maintain contact with a physical surface [131]. Instead, the robot aims a laser pointer at a distant fixed point, which is

safer for the robot and can increase the accuracy. Recently, Tan et al. used the method to calibrate a micropositioner with the assistance of a microscope vision system [132].

5.3.4.1 Limitations of the Point-Contact Method

One limitation of the point-contact method (and other closed-loop techniques) is that it does not work without a precisely known length value [110]. Mathematically, this is due to a linear dependency among the length parameters in the identification Jacobian (section 5.3.2.2). The solution used in [110] was to assume that one of the length parameters was known. However, this may not be a good practical solution because the reason that the calibration is being performed is because accurately known parameters are not available. Furthermore, if the known length parameter happens to be linearly dependent on some other parameter (due to the geometry of the robot), the calibration will still not work

As pointed out in [109], a robot with accurate feedback from a prismatic joint can provide the requisite knowledge of length. If the robot lacks a prismatic joint or feedback, some form of external measurement pertaining to a distance moved by the end-effector can be made. In this case, some simple technique for obtaining the measurement is desired.

Another limitation mentioned in [110] has to do with the orientation of the end-effector. Since only the end-effector's position is constrained in point-contact calibration, the orientation of the end-effector is unknown. Consequently, the paper points out that the link twist parameter (from the standard form of the DH convention) of the final joint (α_{n_j}) cannot be identified during calibration. The result is that the orientation of the end-effector will remain twisted according to the error in the original model, even after the calibration.

Measurements or constraints on the end-effector's orientation would be needed to correct this problem. Unfortunately, this would require external equipment or a more difficult calibration

procedure, which is not desirable. Therefore, it is reasonable to accept the twist of the original model if the twist is not especially important for the intended application (e.g., positioning a cylindrical Raman probe).

It should be noted that the unknown orientation parameter changes if a different robotics convention is used (e.g., Craig's modified form of the DH convention). In addition, other parameters may be affected if they are linearly dependent on the unknown orientation parameter. A method to detect the affected parameters is desired.

As stated in section 5.3.3, closed-loop techniques only (attempt to) calibrate the model with respect to its base coordinate system. This is due to the lack of measurements from an external device in some external coordinate system. Therefore, the robot cannot move with respect to data in an alternate coordinate system (such as that of an imaging device) without an appropriate transformation. This transformation would essentially locate the position and orientation of the robot's base in the alternate coordinate system. A registration procedure could be used to provide this transformation.

In [110], there appears to be the implicit assumption that by including the loop-closing transformation (section 5.3.3.1), the exact location of the end-effector can be determined with respect to the base of the robot's model. Their results (Table VI of [110]) suggest that the technique correctly identified all of the DH parameters (excluding the known length parameter, the final twist parameter, and the arbitrary parameters of the loop-closing transformation). However, experiments have shown that there is another class of parameters that cannot be identified with the point-contact method.

The experiments (described below) indicate that any parameter that has a uniform effect on the end-effector's constrained pose for all joint configurations is not identifiable. For example,

consider the link offset parameter of the first joint (d_1) when the joint is revolute and the model uses the standard form of the DH convention. If d_1 increased by 1 cm, the calculated pose for all joint configurations will be extended by exactly 1 cm. Thus, the change in d_1 produces no relative differences among the computed end-effector locations for the joint configurations. Since point-contact calibration only modifies the model parameters to minimize relative differences in the calculated positions, no modifications will be made for the change in d_1. Hence, d_1 cannot be identified by the technique.

Modifying d_1 is equivalent to translating the entire robot (moving its base). Since the closed-loop technique incorporates no absolute measurements of the end-effector's pose with respect to its base (only joint values), it has no way of knowing where the robot's base should actually be located. The parameters of the loop-closing transformation (which is supposed to go from the end-effector to the robot's base) are variables that are included in the calibration. Therefore, if d_1 is modified to extend all calculated poses by 1 cm, the loop-closing variables can simply increase to include that additional 1 cm. Again, d_1 will not be identified.

Continuing with the same example, the same problem will occur for the first joint angle offset (θ_1^{off}). Modifying θ_1^{off} is equivalent to rotating the entire robot (spinning its base). Any change in θ_1^{off} will produce no relative differences among the computed end-effector locations for the joint configurations. Thus, θ_1^{off} will not be identifiable.

To demonstrate the issue, a simulation experiment was performed with the robot model used for the point-contact example in [110]. First, 30 joints configurations were generated that caused the end-effector of the robot to reach an arbitrary fixed point from various orientations. Then, using their initial model parameters (from Table V), a closed-loop calibration algorithm comparable to theirs was run on the data. Everything was simulated without introducing any

noise/error into the process. The actual/true model and the resulting calibrated model are shown below in Table 5.1.

	Joint, i	d_i / s_i (m)	a_i (m)	α_i (rad)	θ_i^{off} (rad)
Point-Contact Robot True Model	0	1.5940	0.7370	3.6000*	0.0000*
	1	1.7220	−0.5270	−0.5030	0.1200
	2	1.0000*	0.0000	0.5300	0.0000
	3	−0.3300	−0.3300	2.3000	0.0000
	4	0.4400	−0.4400	−0.9000	0.0000
	5	−0.6600	−0.5500	1.2000	0.0000
	6	−1.7930	0.9110	−1.8600*	1.8250
	Joint, i	d_i / s_i (m)	a_i (m)	α_i (rad)	θ_i^{off} (rad)
Point-Contact Robot Calibrated Model	0	1.5512	0.7547	3.6000*	0.0000*
	1	1.6836	−0.5270	−0.5030	0.1454
	2	1.0000*	0.0000	0.5300	0.0000
	3	−0.3300	−0.3300	2.3000	0.0000
	4	0.4400	−0.4400	−0.9000	0.0000
	5	−0.6600	−0.5500	1.2000	0.0000
	6	−1.7930	0.9110	−1.8600*	1.8250

Table 5.1: Denavit–Hartenberg parameters of the point-contact robot [110] for its true model and a calibrated model generated from ideal simulation data. The parameters marked with an asterisk were excluded from the calibration. The calibrated parameters shaded in gray do not match the true parameters.

In Table 5.1, joint 0 represents the loop-closing transformation. Since it is not an actual joint, the variable portion of the joint value (θ_0^{var}) was assumed to be fixed at 0 rad. It is clear from the results that the calibrated values of d_1 (s_1 in their notation) and θ_1^{off} do not match the true values. To verify the results, some forward kinematics calculations (section 2.3.1.2) were performed.

To demonstrate that valid joint configurations were used, the (closed-loop) position of the end-effector was calculated for all 30 joint configurations using the true model. Similarly, to verify that the implementation of the calibration algorithm was correct, the calculations were repeated using the calibrated model. Per the definition of point-contact calibration, all the calculated positions should be (0, 0, 0) if the data represented a fixed point and the calibration

algorithm operated ideally. This is shown for two example joint configurations (from the thirty) in Table 5.2, where $\boldsymbol{\theta}^j$ contains $[\theta_1^{\text{var}}, \theta_2^{\text{var}}, \theta_3^{\text{var}}, \theta_4^{\text{var}}, \theta_5^{\text{var}}, \theta_6^{\text{var}}]$:

- $\boldsymbol{\theta}^1 = [0.7019,\ 1.1987,\ -1.5348,\ -1.0081,\ -1.1480,\ -0.6245]$ rad
- $\boldsymbol{\theta}^2 = [1.3863,\ 0.0161,\ 0.2820,\ 0.2294,\ -1.9752,\ -0.8118]$ rad

Joint Configuration	Closed-Loop Position of True Model (Joints 0–6)	Closed-Loop Position of Calibrated Model (Joints 0–6)
$\boldsymbol{\theta}^1$	(0.0000, 0.0000, 0.0000) m	(0.0000, 0.0000, 0.0000) m
$\boldsymbol{\theta}^2$	(0.0000, 0.0000, 0.0000) m	(0.0000, 0.0000, 0.0000) m

Table 5.2: Computed closed-loop end-effector positions for two example joint configurations using the true and calibrated models of the point-contact robot.

Next, the loop-closing transformation (joint 0) was removed from the true and calibrated robot models since it is a mathematical construct that is not a part of the actual robot. Then, the (open-loop) end-effector position was calculated for all 30 joint configurations. If the calibration were correct, the positions calculated with the true model would match those of the calibrated model. However, as shown for the two example joint configurations in Table 5.3 (and the others), they did not.

Joint Configuration	Open-Loop Position of True Model (Joints 1–6)	Open-Loop Position of Calibrated Model (Joints 1–6)
$\boldsymbol{\theta}^1$	(−0.7370, 0.7054, 1.4294) m	(−0.7547, 0.6864, 1.3910) m
$\boldsymbol{\theta}^2$	(−0.7370, 0.7054, 1.4294) m	(−0.7547, 0.6864, 1.3910) m

Table 5.3: Comparison of the computed open-loop end-effector positions for two example joint configurations using the true and calibrated models of the point-contact robot.

The fact that the error metric of the point-contact calibration (deviation of the closed-loop positions from the origin) was zero but the actual open-loop position was incorrect demonstrates two things. The first is that the point-contact calibration algorithm was implemented and executed correctly, and the second is that the algorithm found incorrect values for d_1 and θ_1^{off}

(along with d_0 and a_0 of the loop-closing transformation). This shows that the point-contact calibration was incapable of identifying those parameters.

In addition, it can be noted that the incorrect parameters aren't too distant from the correct ones. By modifying the initial model or implementation details of the algorithm, an unlimited number of incorrect values that are arbitrarily close to the true values can be produced (while maintaining zero closed-loop error). This is further evidence that those parameters are unidentifiable.

Finally, the set of unidentifiable parameters can change depending on the convention used for the robot's model (e.g., Craig's modified DH convention). In addition, any parameters that are linearly dependent on the unidentifiable parameters will also be unidentifiable. As explained under section 5.5, several conditions can cause linearly dependent parameters (e.g., robots with prismatic joints). Therefore, it is important to have a technique that can recognize and manage the unidentifiable parameters.

The simulations in [110] were only run under ideal conditions. The models had no ambiguous parameters (e.g., due to prismatic joints or consecutive parallel joint axes), and perfectly accurate values were used for parameters that were recognized as unidentifiable. In addition, no noise or error was included in their simulated measurements. Thus, their results were not as meaningful since problems that can occur in more realistic scenarios were not tested. A more complete testing of the point-contact technique would be desirable.

5.3.4.2 Benefits of the Proposed Technique

The fixed-point calibration technique described in this chapter has a number of beneficial features. It was designed to overcome or mitigate some of the weaknesses in alternative techniques. In addition, it was made to be as simple as possible for the user to execute.

The proposed technique includes an algorithm for the detection of unidentifiable parameters (section 5.4.4.2). The algorithm automatically (without user intervention) detects and excludes from calibration any parameters that cannot be identified by the fixed-point technique. This includes the last link's orientation parameter, the parameters that have a uniform effect of the end-effector's constrained pose, and parameters that are linearly dependent on those parameters. The parameters are detected for both forms of the DH convention used in this dissertation. In addition, the algorithm could be easily applied to other conventions because it does not require the calculation of any Jacobian matrices.

The algorithm for detecting unidentifiable parameters is integrated in the core of the calibration technique. It is repeatedly executed as the parameters are calibrated. This is important because the set of unidentifiable parameters depends on the true geometry of the robot, which is not perfectly known. By iteratively checking for unidentifiable parameters, the result is more likely to be correct as the model being calibrated approaches the true model. Note that due to noise/error and complexities in the model, the results may not be perfect. However, experiments indicate that it works rather well (section 5.7).

As mentioned previously, closed-loop calibration cannot always determine the scale of the robot, and certain parameters that determine the position/orientation of the robot's base may be unidentifiable. In addition, closed-loop calibration has no way to locate the robot's base relative to some external coordinate system. To simultaneously overcome all of these issues, the proposed technique includes a targeted registration process (section 5.4.5). By performing this registration, the scale of the robot is determined and a transformation from the robot's base coordinate system is found so that the robot can operate in an external coordinate system. The

registration doesn't actually find the true values of the unidentifiable parameters related to the base, but the registration transformation compensates for their combined error.

The registration process requires the coordinates of registration points that have been externally measured. While this is not ideal, it is an unavoidable necessity due to the limited measurement data of closed-loop calibration. Moreover, a registration would be needed in any case where the robot must operate in an external coordinate system, such as that of an imaging device. By solving multiple problems with a single registration, the overall process is much simpler for the user.

The calibration technique allows multiple fixed points to be used instead of just a single point (section 5.4.3). This provides two advantages. The first is that it may be easier to achieve a sufficient number of joint configurations. The constraints of closed-loop calibration can make it difficult to achieve diverse poses of the end-effector, but alternate fixed points can offer new possibilities for joint configurations. The second advantage is that using multiple points throughout the robot's working envelope may improve the observability of the parameters and accuracy of the calibration [109, 111].

The implemented parameter calibration algorithm (section 5.4.4) uses a bounded direct search technique. It only uses the forward kinematics equations and does not require the computation of any Jacobian matrices. Consequently, it would be easier to adapt the technique to work with other robot modeling conventions. The technique also appears to be rather robust with respect to parameter uncertainties and numerical issues due to the parameter bounds and direct usage of the forward kinematics equations. The direct search makes the technique somewhat slower, but this is not a concern for general robot calibration.

Finally, the presented calibration technique has been extensively tested through a variety of experiments (section 5.6). The experiments included ideal and noisy simulations of four different robot models. These models included ambiguous parameters, and unidentifiable parameters were given inexact values. In addition, experiments were performed on two real robots to evaluate how the calibration algorithms perform under realistic conditions.

5.4 Description of the Technique

5.4.1 Overview

The developed kinematic calibration technique is divided into four phases. Two of the phases require positioning the robot and acquiring measurements, and the other two phases involve the execution of software algorithms. Figure 5.1 provides an overview of the four phases, including their inputs and outputs. The following sections describe the entire calibration technique in detail.

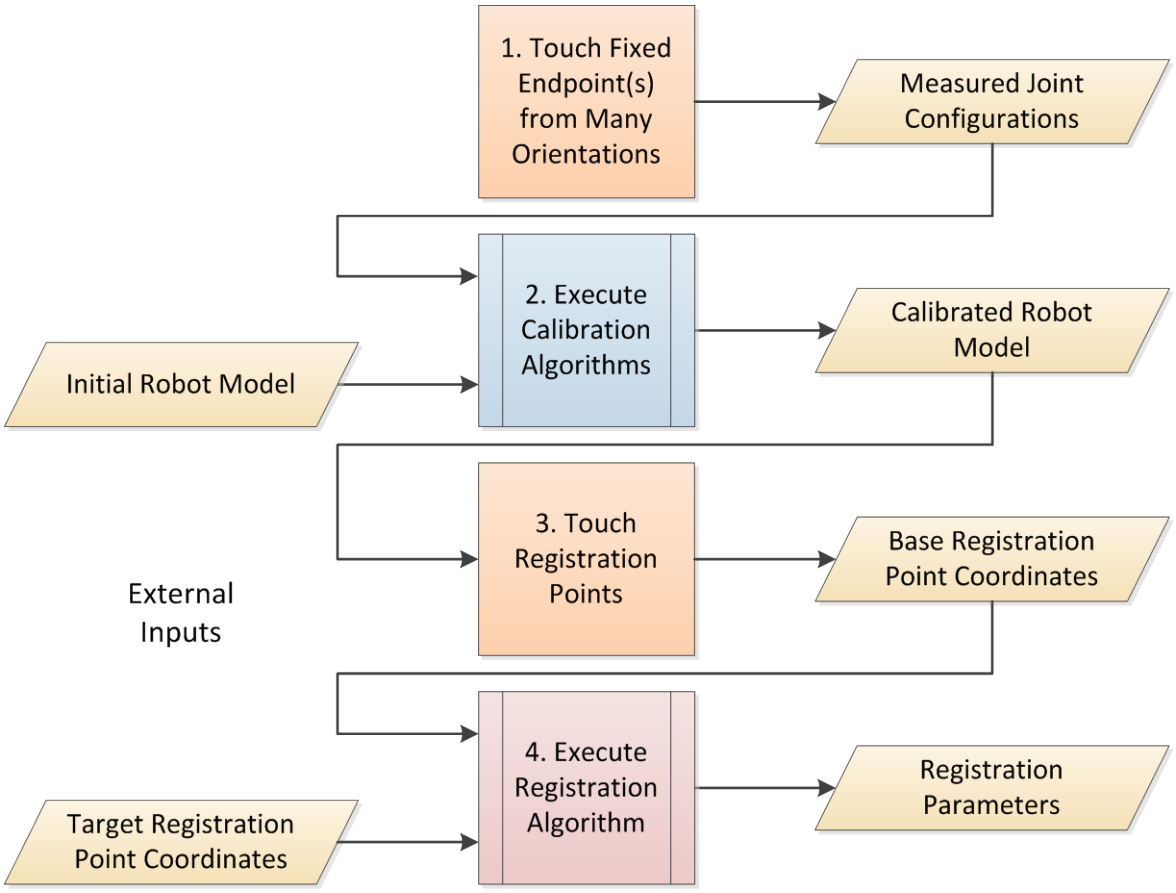

Figure 5.1: Overview of the four phases of the developed kinematic calibration technique.

5.4.2 External Inputs to the Technique

As shown of the left side of Figure 5.1, there are two external inputs required by the calibration technique (the initial robot model and target registration point coordinates). Unlike the other data, they are not generated during the execution of the calibration technique. This section provides a summary of these inputs and how they can be obtained.

The first external input, the initial DH model of the robot, is extremely important. It provides the overall structure of the robot and serves as a starting point for the calibration algorithms. If the initial model does not correctly portray the structure of the robot or contain reasonable estimates of the parameters (per section 5.4.4.3), the calibration will probably fail.

In the implemented version of the calibration technique, the model must be expressed in either the original form or Craig's modified form of the DH convention. The joints must be prismatic (linear) or revolute, or an immobile (fixed) joint can be specified. All four DH parameters must be specified for each joint (with reasonable accuracy), including any fixed offsets for the variable joint parameters (θ_i^{off} or d_i^{off}).

Ideally, the initial model of the robot can be obtained from the manufacturer. The manufacturer should have access to the component schematics of the robot, from which relatively accurate DH parameters can be produced. The errors in a DH model from a manufacturer should only be the result of limitations in the manufacturing/assembly process or wear to the device.

If an initial model is not available, one can be generated with some effort. First, it is necessary to create a general DH model of the robot that encompasses all the joints, links, coordinate frames, etc. This process can be found in a book on robotics, such as [28]. Next, the lengths and angles of the robot must be carefully measured to assign relatively accurate values to the DH parameters. This can be accomplished using common measurement tools, such as a caliper and protractor. Some ingenuity or disassembly may be required to measure parts of a robot with a complicated structure.

The second external input is the target registration point coordinates. These points represent the fixed locations on some object with which the robot must be registered. The object must be rigid (not deformable) to ensure that the registration points are in known locations. For example, indented points on a hard plastic skull were used for testing the image guidance system in Chapter 4.

The coordinates of the registration points should be expressed in some chosen target coordinate system. The target coordinate system is specific to chosen application, and it doesn't have to be similar to the robot's base coordinate system. For instance, a starting position on the surface of a spacecraft could be designated as the origin of a target coordinate system for an outer space application. For environments that have been scanned with an imaging device (e.g., a human patient), the coordinate system of the imaging devices can be used.

There are different ways by which the coordinates of the registration points can be obtained in the target coordinate system. It could involve precise measurement of a target object using a device like a coordinate measuring machine. If accurate imaging of the object has been performed, the registration point coordinates can be derived from the imaging data. For example, the coordinates of the registration points on the phantom skull of Chapter 4 were obtained by locating the indented points in CT scan data of the skull. In this case, the CT imaging data both defined the target coordinate system and provided the coordinates of the registration points.

At least three registration points must be available. The registration points are used to find seven unknown parameters, and three equations can be written for each registration point. Additional registration points are preferred to help reduce the effects of error when attempting to position the robot's end-effector at the registration points.

5.4.3 Phase 1: Measurement of Joint Configurations

The developed calibration technique does not rely on measurements from an external device, such as a laser interferometer or stereo camera system. Instead, it only uses measurements of the joint values (angles or lengths) of the robot. Most robots have built-in sensors (e.g., encoders)

that provide this information since it is typically used to move and track a robot. Hence, the measurements can generally be performed without modifying the robot.

When the end-effector of the robot is held in a particular pose, all of its joints have specific values that enable the end-effector to achieve the pose. The variable value of a revolute or prismatic joint is denoted by θ_i^{var} or d_i^{var}, respectively. As mentioned before, a set of joint values that produces a particular pose of the end-effector will be referred to as a *joint configuration*. The number of values in a complete joint configuration is equal to the number of joints in the robot.

As indicated in Figure 5.1, the presented calibration technique requires the measurement of multiple joint configurations that enable the end-effector to reach a particular fixed point from different orientations. In other words, the tip of the end-effector must be kept in a fixed position while the robot's joints are manipulated to achieve different approaches to the fixed position.

If the robot can be moved by hand, it can be relatively simple to generate the required joint configurations. One simply has to hold the end-effector at a fixed point (e.g., a mark on a flat surface), repeatedly alter the robot's joints (while maintaining contact with the fixed point), and ask the robot's control system to record the joint values. This process can be executed using only the human hand and eye.

If the robot cannot be moved by hand, it is more difficult to generate the required joint configurations. The robot cannot simply be commanded to move to the fixed point because the kinematic calculations would be using the uncalibrated model, which would lead to inaccurate positioning. An iterative approach could be used in which the operator makes small corrections that compensate for the error, although this would be time consuming. Alternatively, the end-effector can be attached to a fixture that functions like a ball joint, keeping the end-effector at a

constant position but allowing the orientation to vary. In either case, force control will be necessary to ensure the robot or fixture is not damaged when the robot is moved. Examples of such control schemes can be found in the literature [113, 128, 133].

In order to approach a fixed point from different orientations, the robot must possess a sufficient number of degrees of freedom. Since rotating about a fixed point uses three passive degrees of freedom, a manipulator with at least four degrees of freedom should work [110]. In some cases, it is possible to use a robot with three degrees of freedom (per section 5.6.2.3 on the PUMA 560), but it is much harder to achieve a sufficient number of poses, and multiple fixed points may be required.

It is important to obtain a sufficient number of joint configurations for the calibration algorithms. The number of equations that can be formed from the measurements must meet or exceed the unknown variables (DH parameters) [62]. The presented calibration technique uses a fixed-point position with varying end-effector orientations, which means that only the (relative) position is known. Since each of the three positional coordinates (x, y, and z) is a function of the DH parameters, three equations can be formed for each measured joint configuration.

A robot with n_j joints has a total of $4n_j$ (unknown) DH parameters. If n_{jc} is defined as the number of measured joint configurations, then the inequality in Equation 5.8 must be satisfied to ensure a sufficient number of joint configurations are obtained. Note that in practical applications, additional configurations should be acquired to reduce the effects of errors and potential dependencies in the data.

$$3n_{jc} \geq 4n_j \qquad \text{Equation 5.8}$$

Finally, the presented technique allows multiple fixed points to be used. There are two reasons for having this feature. First, it may be physically easier to acquire joint configurations from multiple fixed points instead of a single one. It some cases (as in section 5.6.2.3), it may even be necessary in order to obtain a sufficient quantity of configurations. Second, choosing points throughout the working envelope may result in a more accurate parameter calibration. A number of techniques have been developed for choosing the best sets of poses for observability of the parameters [109, 111].

5.4.4 Phase 2: Execution of the Calibration Algorithms

After a sufficient number of joint configurations have been captured from fixed point(s), the next phase entails the execution of the kinematic calibration algorithms. As indicated by Figure 5.1, these algorithms use the measured joint configurations and an initial DH model of the robot to produce a calibrated model of the robot. Collectively, the calibration algorithms can be classified as *offline*, which means that all the input data must be available before execution.

There are a total of three algorithms used for the calibration process. The main algorithm is called the high-level calibration algorithm because it runs the other algorithms. The other algorithms are the effective parameter identification algorithm and the parameter optimization algorithm. All three of the algorithms are described in the following sections.

5.4.4.1 High-Level Calibration Algorithm

The high-level calibration algorithm is summarized in Figure 5.2. Its purpose is to generate a calibrated model of the robot using an initial model and the recorded joint configurations. The high-level algorithm iteratively executes the other two algorithms until a suitable calibrated model is produced.

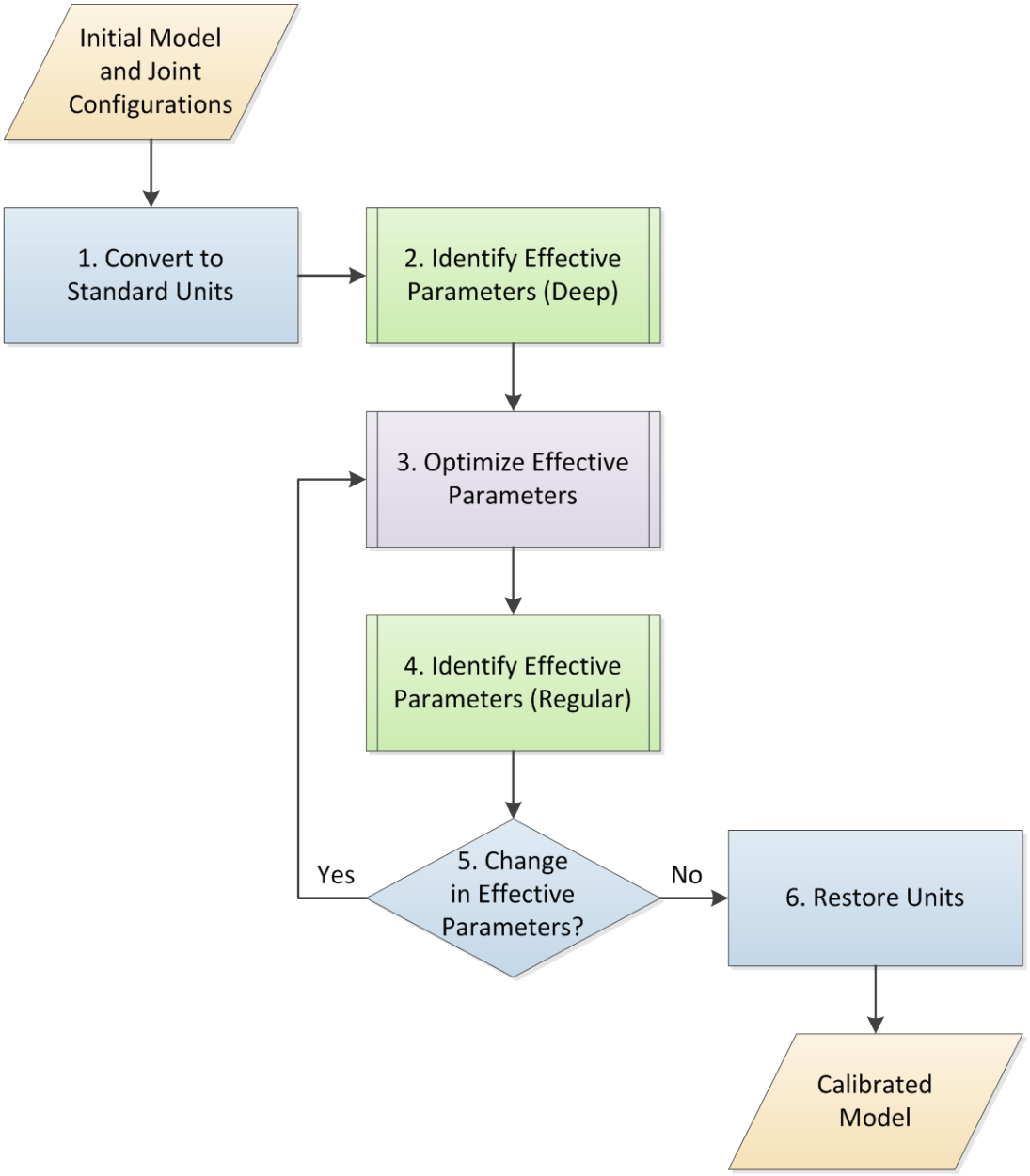

Figure 5.2: Diagram of the high-level calibration algorithm.

Step 1 of the algorithm is to convert the initial model and joint configuration (pose) data to standard units of measurement. For simplicity, lengths are converted to meters, and angles are translated to radians. There are more complicated ways to scale the DH parameters and end-effector pose values for potentially better performance [111], but the simple conversion was

found to be sufficient. In addition, the technique's focus on position error means that the units do not have to be scaled to make orientation error comparable to position error.

In step 2, the effective parameter identification algorithm is executed to generate an initial list of effective DH parameters. *Effective parameters* are defined as those that have an effect (when modified) on the chosen error metric, and thus can be calibrated. (Section 5.4.4.2 has more details.) The effective parameters are identified so the optimization algorithm only tries to alter parameters that have a chance of improving the model. In fact, attempting to modify the ineffective parameters is likely to result in a worse model because the algorithm cannot detect when the changes have a negative effect.

For the initial list of effective parameters, the identification algorithm is run in "deep" mode. This means that the algorithm attempts to find the effective parameters using not only the initial DH model, but also using modified versions of the DH model (explained in section 5.4.4.2). This is done because the starting model is uncalibrated and thus cannot be trusted to produce the ideal list of effective parameters. The goal of the initial deep search is to find all the parameters that could potentially be effective. This list may ultimately contain too many parameters, but the largest list is desired for the first attempt at calibration.

For step 3, the parameter optimization algorithm is run to calibrate all of the identified effective parameters. This algorithm is described in section 5.4.4.3. The accuracy of the calibrated parameters depends upon the accuracy of the effective parameter list (among other things). Hence, it is important to recheck the effective parameter list, as explained below.

In step 4, the effective parameter identification algorithm is executed again on the newly calibrated model. This time, the identification is run in "regular" mode. In this mode, the algorithm finds the effective parameters using only the calibrated model (not any modified

versions of it). Therefore, the generated list of effective parameters represents the best estimation (so far) of the list of parameters that should have been calibrated. Note that due to noise and other sources of error, it may not be possible to find the ideal list of effective parameters.

For step 5, the new list of effective parameters is compared against the previously calculated list of effective parameters. If there are any differences, it means that the parameter optimization algorithm probably did not calibrate the ideal list of parameters and thus did not generate the best possible calibration. In this case, the high-level algorithm returns to step 3 to make another attempt at calibrating the robot's model using the new list of effective parameters.

If there was no change in the list of effective parameters, it is assumed that the parameter optimization algorithm ran on the correct list of effective parameters and therefore produced the best possible calibration. This is the desired case, which may take a few iterations to achieve. Consequently, the high-level algorithm proceeds to step 6.

It is possible for the new list of effective parameters to be different from the immediately previous list but match some preceding list. If left unhandled, this would lead to a cycle and hence an infinite loop. Thus, there is some additional logic to prevent cycles from occurring in the lists of effective parameters. If a repeat list is produced, the algorithm computes the intersection (from set theory) of the list(s) produced after the original matching list. It then repeats the comparison of step 5 to determine how to proceed.

In step 6, the units of measurement of the final calibrated model are restored to the original units. This makes the previous conversion of units invisible to the user. With this step, the high-level calibration algorithm is complete.

The final calibrated model (along with the set of registration parameters) is an important output of the overall calibration technique. It serves as the replacement for the initial uncalibrated model, and it is used in the next phase of the technique. If everything goes as intended, the calibrated model should provide higher accuracy than the initial model when combined with the registration parameters.

5.4.4.2 Effective Parameter Identification Algorithm

The goal of the effective parameter identification algorithm is to identify a list of model parameters that have an effect (when modified) on the error metric chosen for the calibration. This needed because of the issues discussed in section 5.3.4.1. This algorithm is called by the high-level calibration algorithm. Its inputs are the current DH model under evaluation, all of the measured joint configurations, and the search type ("deep" or "regular"). The algorithm uses this data to produce an effective parameter list. An overview of the algorithm is shown in Figure 5.3.

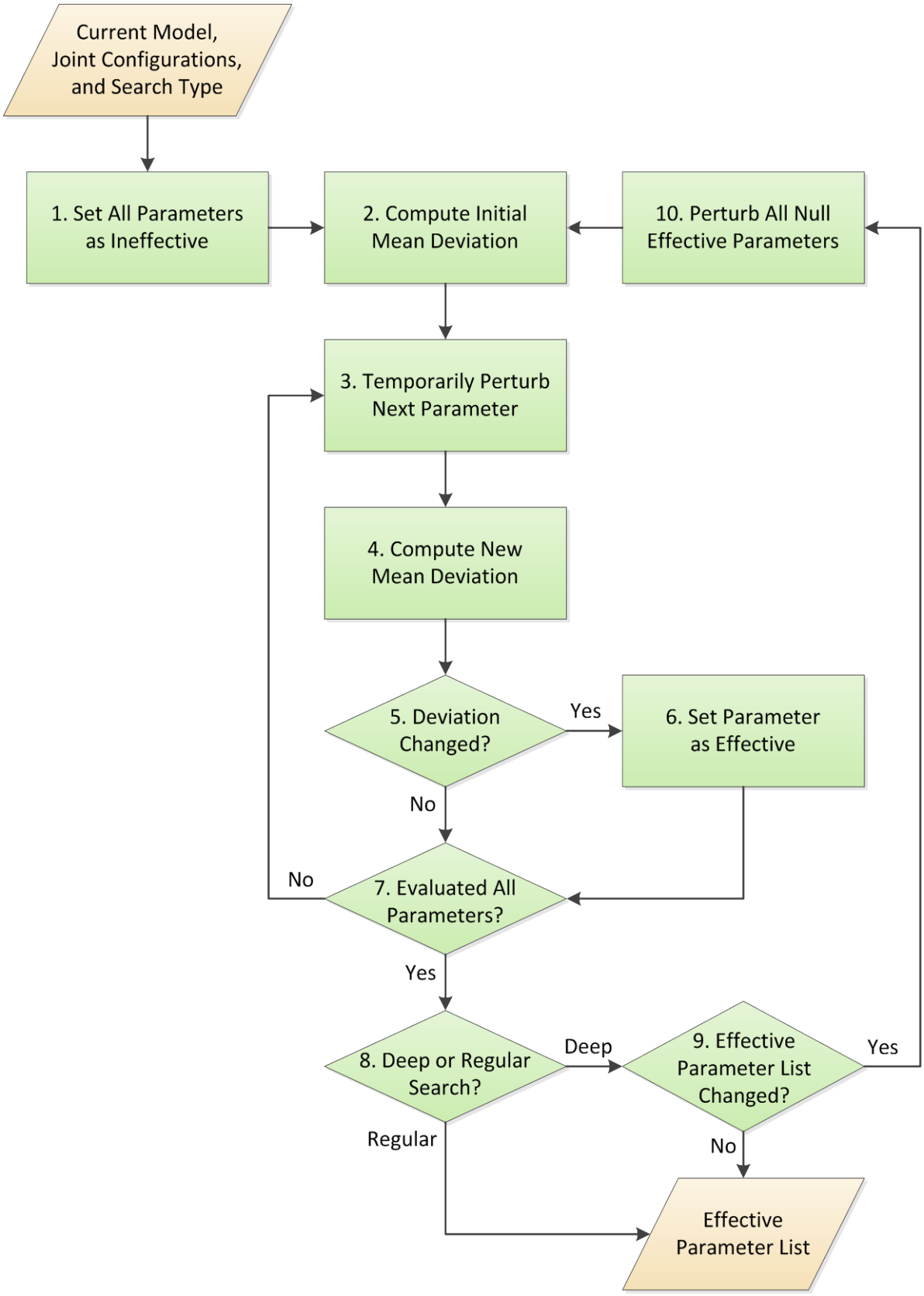

Figure 5.3: Diagram of the effective parameter identification algorithm.

For step 1, the algorithm begins by assuming that all the parameters are ineffective. In step 2, the value of an error metric is computed for the current DH model. The chosen error metric is the mean deviation of the end-effector from the (estimated) fixed-point location(s). It is calculated in two stages.

First, each fixed-point location is estimated (because the technique uses no absolute external measurements). This is accomplished by computing the mean position of the end-effector when all the joint configurations for a specific fixed point are run through the forward kinematics algorithms (section 2.3.1.1) using the current DH model. Equation 5.9 shows the formula.

$$\widehat{\boldsymbol{p}}_{\mathrm{ep},i} = \frac{\sum_{k=1}^{n_{\mathrm{jc},i}} \boldsymbol{p}_{\mathrm{ee},i,k}}{n_{\mathrm{jc},i}} \qquad \text{Equation 5.9}$$

The following terms are used in Equation 5.9:

- $\widehat{\boldsymbol{p}}_{\mathrm{ep},i}$ is the estimated location (position) of fixed point i
- $n_{\mathrm{jc},i}$ is the number of joint configurations (poses) measured for fixed point i
- $\boldsymbol{p}_{\mathrm{ee},i,k}$ is the position of the end-effector (computed using forward kinematics) for the k^{th} joint configuration associated with fixed point i

Second, the mean deviation of the end-effector's position from the estimated fixed-point locations is determined. This is achieved by calculating the end-effector's position for each joint configuration (using the forward kinematics equations with the current DH model) and computing the Euclidean distance from the end-effector to the corresponding fixed point. Equation 5.10 and Equation 5.11 represent this process.

$$d_{i,k} = \left\| \widehat{\boldsymbol{p}}_{\mathrm{ep},i} - \boldsymbol{p}_{\mathrm{ee},i,k} \right\| \qquad \text{Equation 5.10}$$

$$\bar{d} = \frac{\sum_{i=1}^{n_{\mathrm{ep}}} \sum_{k=1}^{n_{\mathrm{jc},i}} d_{i,k}}{n_{\mathrm{jc}}} \qquad \text{Equation 5.11}$$

The following new terms are used in the above equations:

- $d_{i,k}$ is the Euclidean distance (deviation) between the estimated location of fixed point i and end-effector pose k of those measured at fixed point i
- $\bar{d}$ is the error metric (the mean deviation of the end-effector from the estimated fixed-point locations)
- n_{ep} is the number of fixed-point locations used for the joint configuration data
- n_{jc} is the total number of measured joint configurations (poses) for all fixed points

For step 3, one DH parameter is temporarily perturbed by a significant value (1000 m or $\pi/3$ rad in the implemented algorithm). The value is large to ensure that any impact on the error metric is also significant. In step 4, the mean deviation is recalculated (per Equation 5.11) using the DH model with the temporarily perturbed parameter. Afterwards, the perturbed parameter is restored to its previous value.

For step 5, the mean deviations computed before and after perturbing the parameter are compared. If the mean deviation changed, it means that the parameter had an effect on the error metric. In this case, the parameter is marked as "effective" in step 6. Otherwise, the parameter is left as ineffective.

In step 7, the algorithm checks to see if all the DH parameters had a chance to be perturbed. If not, the algorithm returns to step 3 to repeat the process with the next parameter. This loop will continue until all the DH parameters of the robot model have been evaluated. Once the loop has finished, a complete list of the effective parameters has been generated.

In step 8, the algorithm checks its inputs to determine the type of search being performed. If it is a "regular" search, no further processing is necessary. Thus, the effective parameter list is

returned to the high-level calibration algorithm, ending the effective parameter identification algorithm. If it is a "deep" search, the algorithm continues to step 9.

For step 9, the newly generated list of effective parameters is compared with the previous list of effective parameters. For the first iteration of a "deep" search, the previous list is considered to be the empty list (all parameters considered to be ineffective). If there are no changes to the list, the algorithm is complete, and the list is returned to the high-level calibration algorithm. If any changes are found, the algorithm continues to step 10.

In step 10, the input DH model is modified to help find more parameters that may be effective. Any null effective parameters (those with a length of roughly 0 m or an angle that is approximately a multiple of 2π rad) are set to a non-null value. The intention is to help find effective parameters that may be hidden by nearby null parameters. For example, a parameter that controls the rotation of a link with an uncalibrated length of 0 m might not appear to have any effect on the end-effector's position. However, if the link's length is actually greater than 0 m, the rotational parameter could be effective. Therefore, any effective parameters with a null value are modified in step 10 to promote the discovery of additional effective parameters.

After step 10, the algorithm returns to step 2 using the modified input DH model. The subsequent steps are run to determine if any more effective DH parameters can be identified. The whole process iteratively repeats until the effective parameter list is found not to have changed in step 9. At this point, the algorithm is finished, and the final list of effective parameters is returned to the high-level calibration algorithm.

Note that due to noise/error and complexities of the model, the identified set of effective parameters may not be ideal. It is possible for some ineffective parameters to be marked as effective and vice versa. To perfectly identify the effective parameters, the true model and

perfect measurements would have to be available. Since this is not a realistic requirement, the algorithm does the best it can with the data available. Mistakes may be mitigated by the fact that is algorithm is run multiple times throughout the calibration process, the calibrated parameters are bounded (section 5.4.4.3.1), and the registration algorithm can compensate for some problems (section 5.4.6).

It should also be noted that steps 1 through 7 are often accomplished in the literature using the identification Jacobian matrix [105, 112]. By performing a singular value decomposition or QR decomposition of the matrix and analyzing the results, it is possible to determine which parameters are unidentifiable, weakly identifiable, or only identifiable in linear combinations of parameters [111]. Section 5.3.2.2 has more details. The presented calibration technique does not require Jacobian matrices to be computed, so the method described in this section for identifying parameters was created. Steps 8 through 10 would remain the same for any identification method.

5.4.4.3 Parameter Optimization Algorithm

The purpose of the parameter optimization algorithm is to perform a calibration of all model parameters identified as effective. This algorithm is called by the high-level calibration algorithm of section 5.4.4.1. It uses the measured joint configurations along with the current DH model and effective parameter list to achieve its goal. A diagram of the algorithm is shown in Figure 5.4.

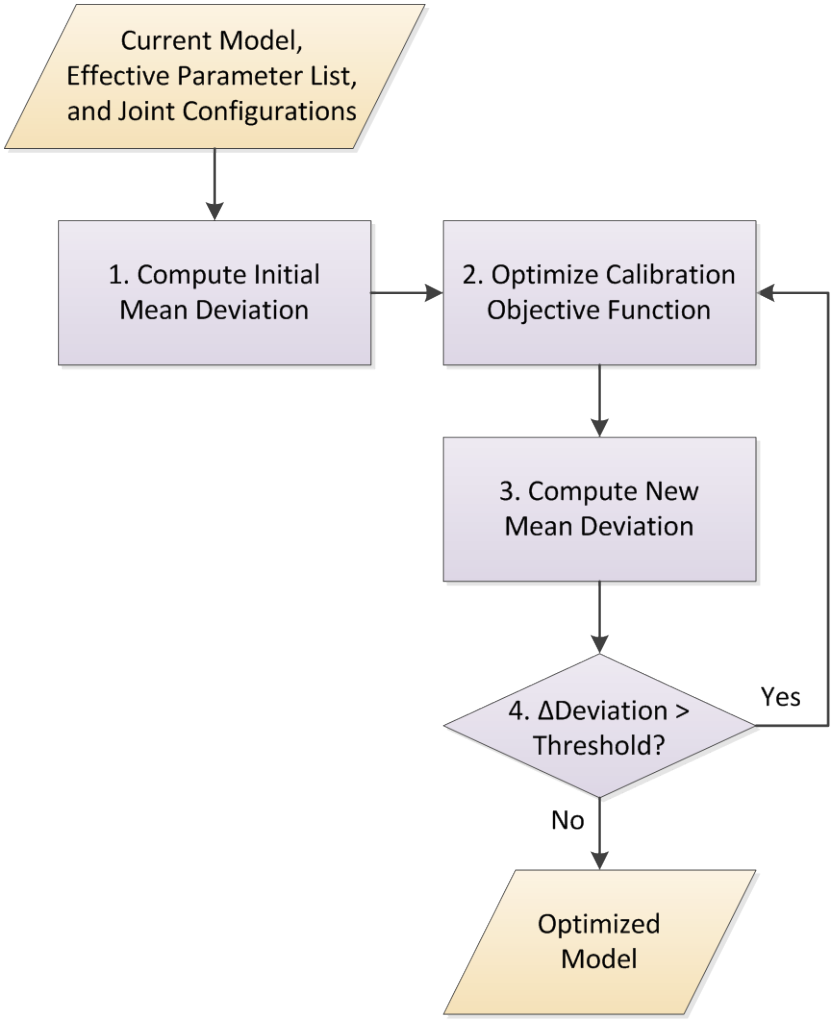

Figure 5.4: Diagram of the parameter optimization algorithm.

Step 1 of the optimization algorithm is to compute an initial mean deviation using the inputted DH model. The calculations are the same as those described in the previous section using Equation 5.11 and its supporting equations. The initial mean deviation is used later in the algorithm to help determine when the optimization is complete.

In step 2, the mathematical optimization of an objective function (f_{cal}) is performed to calibrate the effective DH parameters. The objective function calculates the previously defined error metric, $\bar{d}$, which represents the mean deviation of the end-effector from the (estimated) fixed-point location(s). Therefore, the objective function encapsulates Equation 5.11 and all its

supporting equations, including the requisite forward kinematics equations (from section 2.3.1.1). Equation 5.12 summarizes the direct relationship between the calibration objective function, f_{cal}, and the error metric, $\bar{d}$.

$$\bar{d} = f_{\text{cal}}(M, C) \qquad \text{Equation 5.12}$$

The following new terms are defined for Equation 5.12:

- M is a set of all the non-variable DH parameters of the robot model (including both the effective parameters, M_{eff}, and the ineffective parameters, M_{ineff}). The parameters that represent variable joint values (θ_i^{var} or d_i^{var}) are contained in the following set.
- C is a set of all the measured joint configurations (the sets of joint values associated with each fixed point).

Equation 5.12 specifies that the mean deviation is a function of the robot model parameters and the joint configurations. Assuming all the joint configurations were captured correctly, the robot model parameters are the primary source of error (deviation). Therefore, the error can be minimized by finding a set of DH parameters that minimizes the objective function. The optimized DH parameters will represent the calibrated robot model.

As explained previously, only the effective DH parameters are intended to be calibrated by this algorithm. Attempting to modify the ineffective parameters would likely increase the pose error produced by the model because the error metric cannot be used to guide their calibration. Therefore, the goal of the optimization is to choose a set of effective DH parameters that minimizes the value of the objective function. Equation 5.13 demonstrates this objective.

$$M_{\text{eff,cal}} = \underset{M_{\text{eff}}}{\arg\min}\, f_{\text{cal}}(M_{\text{eff}}, M_{\text{ineff}}, C) \qquad \text{Equation 5.13}$$

In Equation 5.13, $M_{\text{eff,cal}}$ represents the calibrated effective DH parameters. They are found using an optimization method that is explained in section 5.4.4.3.1. The calibrated effective parameters, $M_{\text{eff,cal}}$, must be combined with the (unoptimized) ineffective parameters, M_{ineff}, to produce a complete set of model parameters. This new set of parameters is the output of step 2.

For step 3, the mean deviation is recalculated (Equation 5.11) using the newly optimized DH model. In step 4, the percentage decrease from the previous mean deviation to the new deviation is computed and compared to a threshold. The default threshold is 1% in the implemented algorithm, but it can be adjusted to control how aggressively the algorithm searches for an answer. If the percentage decrease in mean deviation is greater than the threshold, the algorithm returns to step 2. In this case, the new DH model is used as the input model, M, for the optimization of the objective function. Otherwise, the algorithm exits and returns the newly optimized DH model to the high-level calibration algorithm.

The reason for the loop between steps 2 and 4 is to help ensure that the optimization method achieves a good solution for the optimized model. The complexity of the objective function can cause the optimization method to produce a solution that is less than optimal if the initial DH model isn't very accurate. One reason for this is the lack of absolute end-effector measurements, which requires the fixed-point locations to be estimated (Equation 5.9 in the previous section). The estimated fixed-point locations depend on the current DH model, and the objective function depends on the estimated locations. Therefore, it is almost like the surface of the objective function changes as the parameters change, which makes the optimization more challenging. This issue is less of a problem when the parameters are more accurate and thus require less change.

Finally, it is important to mention that the objective function, like the overall calibration technique, only takes into account the mean positional deviation of the end-effector. The measured data does not enable any calculation of error in orientation. Thus, although the final model should improve the positional accuracy of the end-effector, some orientation error is likely to remain.

5.4.4.3.1 Optimization of the Calibration Objective Function

Finding an optimized set of effective parameters (Equation 5.13) can be a difficult task. The number of effective parameters is typically large (up to four per joint), which means the search space for a solution has a high dimension. In addition, the objective function (Equation 5.12) is a complicated nonlinear function. Consequently, there is no simple equation that will produce the desired set of parameters. Instead, a method based on mathematical optimization or nonlinear estimation can be used to find a solution.

In the implemented algorithm, a standard optimization technique, the Nelder–Mead simplex method [76], is used to optimize the objective function. This method constructs a simplex (the generalization of a triangle to higher dimensions) in the parameter space. If n_{eff} is the number of parameters being optimized, then the simplex consists of $n_{\text{eff}}+1$ vertices. Each vertex represents a possible set of effective parameters. Through iterative evaluation of the objective function at numerous test points (sets), each vertex is slowly moved in the direction of decreasing error. Eventually, the simplex converges about a point that represents the optimized set of effective parameters.

The simplex method does not require any gradients or Jacobian matrices to be calculated, which makes it easier to adapt to different robot conventions. Other calibration techniques often use a gradient-based method like the Gauss–Newton or Levenberg–Marquardt algorithms to find

the solution [105, 111]. In general, these methods converge much faster, but they require Jacobian matrices to be computed. In addition, a direct search algorithm like the simplex method may be more forgiving of numerical issues than the gradient-based methods [105].

It is possible for the chosen optimization technique (or any other) to produce incorrect or unexpected solutions when trying to minimize the objective function. The resulting robot models may differ from the desired model in subtle or significant ways. There are at least five different causes for this issue, which are summarized below.

First, the presence of noise and other sources of error in the joint measurements will prevent the optimization from generating an ideal solution. The accuracy of any calibration depends upon the quality of the data on which it is based. Special care should be taken during the measurement of joint configurations to avoid introducing any additional noise/error.

Second, the fact that parameters marked as "ineffective" are not calibrated will almost certainly lead to an inexact solution. Unless the ineffective parameters had perfect values from the start, their values in the final model will remain imperfect. The registration process attempts to compensate for these deviations, but it still means that the calibrated DH model is not ideal. In addition, there will be further error if any parameters are improperly identified as "effective" or "ineffective". The list of effective parameters is rechecked after every optimization to help prevent this from happening.

A third cause of alternate solutions is the fact that a DH model is often not an ideal, minimal representation of a robot. As explained under section 5.5, there can be poorly defined or redundant DH parameters, which means there are multiple sets of parameters that effectively produce the same robot. For example, the presence of prismatic joints or consecutive parallel joint axes will lead to such problems. Therefore, the optimization method is likely to find one of

the alternative solutions if the robot's structure has any of these issues. Even though the solution might not exactly match the physical structure of the robot, the positional accuracy of the end-effector will be the same as that for all of the equivalent models.

Fourth, the optimization method might find a trivial (but unrealistic) solution for minimizing the objective function. For example, the error can be zero if all the length parameters are set to 0 m or all the joint axes are made parallel. To avoid these kinds of incorrect solutions, the search space for M_{eff} can be constrained to small region around the initial parameter values of M_{eff}. This remedy assumes a reasonable initial model of the robot is available (which is required by the calibration technique).

Fifth, non-ideal solutions may be produced if the optimization method becomes stuck at a local minimum of the objective function. In general, optimization methods cannot always achieve the ideal, global minimum value of an objective function. To reduce the chance of finding a local minimum, the optimization method should start its search from a reasonable estimate of the solution. In this case, the starting point can be the initial DH model of the robot. A kinematic model based on the DH convention is only mildly nonlinear [111], so a good initial model should help the optimization method achieve the correct solution.

Therefore, to help the optimization method achieve the best possible solution, two measures are taken with respect to its execution. First, the initial model of the robot is used for the starting parameters of the optimization. This also reduces the time needed to find the solution. Second, each parameter being optimized is given an upper bound and a lower bound based on the corresponding initial parameter. The optimization method will not exceed the specified bounds while searching for optimal parameter values. For the implemented algorithm, these bounds

default to ±2 cm for length parameters and ±0.3 rad for angular parameters (with respect to the initial parameters).

The bounds for the parameters can be adjusted to suit the accuracy of the initial model. For a more precise model, smaller bounds can be used to increase the speed of the search. However, the bounds must be at least as large as the maximum error in any of the initial DH parameters, or the correct solution will fall outside the optimization method's search space. For less precise models, larger bounds must be used. Note that if the bounds are too large, non-ideal solutions are more likely to be found.

With these measures in place, the optimization was found to be sufficiently robust. For example, it was able to handle robot models with parallel or nearly parallel consecutive joint axes. Commonly, an alternative robotics convention (such as one based on Hayati's system [134]) is used to avoid numerical issues that occur with parallel axes (section 5.5). All of the experiments in this chapter (section 5.6) were successfully executed on challenging robot models using only standard DH conventions.

5.4.5 Phase 3: Measurement of Registration Points

As shown in Figure 5.1, the coordinates of registration points are used by the presented technique. The coordinates of these points need to be known in two different coordinate systems. The first is the target coordinate system, which is used to define positions in the robot's operating environment. The second is the robot's base coordinate system, which is specific to the robot model being used.

The registration point coordinates in the target coordinate system are assumed to be an external input to the system, as explained in section 5.4.2. The coordinates in the robot's base coordinate system must be acquired during this phase of the calibration technique. Once the

coordinates of the registration points are available in both systems, the registration algorithm can generate the registration parameters (section 5.4.6). The parameters enable the robot to operate in a target coordinate system instead of being limited to the robot's base coordinate system.

The process of obtaining the registration point coordinates in the base coordinate system is very similar to the method used to measure the fixed-point joint configurations in section 5.4.3. First, the user must place the end-effector of the robot at one of the registration points on the object to which the robot is being registered. This can be accomplished via manual positioning or a more complicated control scheme. Once the end-effector is accurately positioned at the registration point, the joint configuration of the robot must be recorded using the robot's control system.

The same process needs to be repeated for each of the remaining registration points. In addition, the identity of the registration point associated with each joint configuration should also be recorded. This will enable the registration algorithm to compare the correct pairs of coordinates between the two coordinate systems.

Finally, using each of the newly recorded joint configurations, the position of the end-effector can be calculated using forward kinematics (section 2.3.1.1). These positions represent the coordinates of the registration points in the robot's base coordinate system. Note that the forward kinematics must be performed using the calibrated robot model from the previous phase. This ensures that the registration algorithm corrects the calibrated model instead of the obsolete initial model.

5.4.6 Phase 4: Execution of the Registration Algorithm

The registration of the robot is executed for two reasons. First, it enables the robot to operate in a desired target coordinate system instead of being limited to the robot's base coordinate

system. For example, this would allow the robot to easily and accurately move to locations determined from imaging data of the environment. Second, the registration compensates for some limitations of the data collected for the calibration process. In effect, it enables the position and orientation of the robot's base and the scale of the robot to be found, which may not be correct in the calibrated DH model.

The inputs to the registration algorithm are the coordinates of the registration points in both the robot's base coordinate system and the target coordinate system. Using this data, it generates a transformation between the two coordinate systems. This transformation is defined by registration parameters that indicate how to translate, rotate, and scale the calibrated robot model to achieve accurate positioning in the target environment. An overview of the registration algorithm is shown in Figure 5.5.

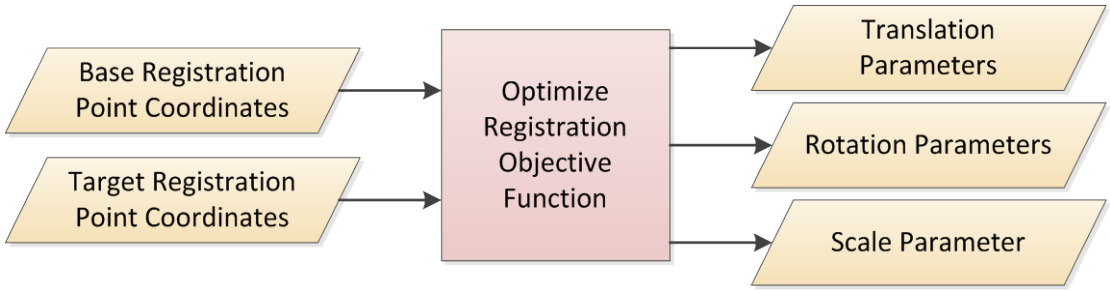

Figure 5.5: Overview of the registration algorithm.

There are a total of seven registration parameters. The three translation parameters (Δx, Δy, Δz) and three rotation parameters (φ, θ, ψ) define the necessary changes in the position and orientation (respectively) of the robot model's base. These parameters conceptually move the entire robot so that its base coordinate system aligns with the target coordinate system. This transformation also compensates for the ambiguous definition of the robot's base in the calibrated DH model, which results from the lack of absolute position measurements in the presented calibration technique. Sections 5.3.3 and 5.3.4.1 have more information.

The translation parameters (Δx, Δy, Δz) represent the distances along the three axes of the base coordinate system that the base of the robot must be moved (conceptually, not physically). The rotation parameters (φ, θ, ψ) describe the overall change in orientation of the robot's base that must occur relative to the base coordinate system. Specifically, the rotation parameters represent Euler angles, which are described in section 2.3.1.1.

The final parameter is the scale parameter, s. This parameter defines the change in the scale (size) of the calibrated robot model that is necessary to achieve accurate positioning. It is a dimensionless quantity that represents the ratio of the new size of the robot model to its original size. This parameter is important because the lack of absolute position measurements in the presented calibration technique can cause the calibrated robot model to have the wrong scale (in some instances). Sections 5.3.3 and 5.3.4.1 have more information.

The goal of the registration algorithm is to generate a specific set of registration parameters. This set will be used to produce a transformation. When the transformation is applied to any point expressed in the robot's base coordinate system, the coordinates of the transformed point should match the coordinates of the same point expressed in the target coordinate system. Hence, the transformation derived from the registration will allow the position of the end-effector (or any frame of the robot) to be known in the target coordinate system. This information will enable the robot to accurately operate within the target environment. The inverse of the transformation will also permit any point in the target coordinate system to be expressed in the robot's base coordinate system.

The registration algorithm only has access to the registration points measured in phase 3 (and those provided as an external input). Thus, it must generate the set of registration parameters (and the corresponding transformation) using only these points. When the transformation is

applied to the registration points expressed in the robot's base coordinate system, the coordinates of the transformed points should match the coordinates of the corresponding points expressed in the target coordinate system.

The mathematical optimization of an objective function (f_{reg}) is performed to generate the necessary registration parameters. The objective function performs three tasks. First, it applies a transformation based on the current set of registration parameters to each registration point expressed in the robot's base coordinate system. Second, it computes the Euclidean distance between each transformed point and the corresponding point expressed in the target coordinate system. Third, it calculates the mean of the distances and returns this value as a measure of the registration error. Equation 5.14 represents the objective function.

$$f_{\text{reg}}(R, \boldsymbol{P}_{\text{b}}, \boldsymbol{P}_{\text{t}}) = \frac{\sum_{i=1}^{n_{\text{rp}}} \left\| \boldsymbol{F}_{\text{trans}}(R)\boldsymbol{p}_{\text{b},i} - \boldsymbol{p}_{\text{t},i} \right\|}{n_{\text{rp}}} \qquad \text{Equation 5.14}$$

The following terms are used in Equation 5.14:

- R is a set of registration parameters: $\{\Delta x, \Delta y, \Delta z, \varphi, \theta, \psi, s\}$
- n_{rp} is the number of registration points
- $\boldsymbol{P}_{\text{b}}$ is a $3 \times n_{\text{rp}}$ matrix that contains column vectors of the positional coordinates of the n_{rp} registration points expressed in the robot's base coordinate system
- $\boldsymbol{p}_{\text{b},i}$ is a 4×1 homogeneous position vector containing column i of $\boldsymbol{P}_{\text{b}}$ with an additional value of '1' in the 4$^{\text{th}}$ location
- $\boldsymbol{P}_{\text{t}}$ is a $3 \times n_{\text{rp}}$ matrix that contains column vectors of the positional coordinates of the n_{rp} registration points expressed in the target coordinate system

- $\boldsymbol{p}_{\mathrm{t},i}$ is a 4×1 homogeneous position vector containing column i of $\boldsymbol{P}_{\mathrm{t}}$ with an additional value of '1' in the 4th location

In Equation 5.14, $\boldsymbol{F}_{\mathrm{trans}}(R)$ is a function that generates a transformation from the base coordinate system to the target coordinate system based on the input set of registration parameters. It is defined by the following equations.

$$\boldsymbol{F}_{\mathrm{trans}}(R) = \boldsymbol{F}_{\mathrm{trans}}(\Delta x, \Delta y, \Delta z, \varphi, \theta, \psi, s) = {}^{\mathrm{b}}\boldsymbol{T}_{\mathrm{t}} = \boldsymbol{T}_{\mathrm{c}}\boldsymbol{T}_{\mathrm{s}} \quad \text{Equation 5.15}$$

$$\boldsymbol{T}_{\mathrm{c}} = \begin{bmatrix} & \boldsymbol{R}_{\varphi,\theta,\psi} & & \begin{matrix}\Delta x\\ \Delta y\\ \Delta z\end{matrix} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{Equation 5.16}$$

$$\boldsymbol{T}_{\mathrm{s}} = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{Equation 5.17}$$

The following terms are used in the above equations:

- ${}^{\mathrm{b}}\boldsymbol{T}_{\mathrm{t}}$ is the matrix representation of a homogeneous transformation from the robot's base coordinate system to the target coordinate system.
- $\boldsymbol{T}_{\mathrm{c}}$ is a homogeneous transformation matrix that combines the change in position defined by $(\Delta x, \Delta y, \Delta z)$ and the change in orientation defined by (φ, θ, ψ).
- $\boldsymbol{R}_{\varphi,\theta,\psi}$ is a 3×3 rotation matrix that is equivalent to the change in orientation defined by (φ, θ, ψ). It is computed using Equation 2.6 from section 2.3.1.1.
- $\boldsymbol{T}_{\mathrm{s}}$ is a homogeneous transformation matrix that represents the change in scale defined by s.

The objective function must be minimized by finding an optimal set of registration parameters, R_{opt}. Equation 5.18 represents this goal. Note that the optimal registration

parameters will probably not produce a perfect transformation due to the noise and other sources of error in the measurement of the registration points.

$$R_{\text{opt}} = \arg\min_{R} f_{\text{reg}}(R, \boldsymbol{P}_{\text{b}}, \boldsymbol{P}_{\text{t}}) \qquad \text{Equation 5.18}$$

Mathematical optimization can be used to find the set of registration parameters that minimizes the objective function. As with the parameter optimization algorithm, the Nelder–Mead simplex method [76] was used. However, unlike the parameter optimization algorithm, no additional iterative procedure is needed. This is because the registration objective function is much simpler than the calibration objective function. In addition, the number of registration parameters (seven) is much smaller than the number of DH parameters for any robot that can be used with the calibration technique.

With the computation of the optimal registration parameters, the kinematic calibration technique is complete. The robot model produced in phase 2 serves as the new, calibrated model of the robot. The transformation provided by the optimized registration parameters enables the robot to work in the target coordinate system and overcomes some limitations of the data collected for the calibration process.

5.5 Caveats for the Technique

There are a few points that the user should be aware of when using the present fixed-point calibration technique. Some of these issues are caused by limitations in the technique, and others are the result of more general challenges in robot calibration. These points are found in the sections below.

5.5.1 Limitations of Denavit–Hartenberg Parameters

In general, DH parameters do not form an ideal or minimal representation of a robot's geometrical structure. A number of issues with the DH convention can be found in literature [28, 63, 105, 111]. Some common problems are described below.

Only two of the four DH parameters are required to specify a prismatic joint. Consequently, there is no one correct way to set the remaining parameters. If these parameters are included in a calibration, their calibrated values could end up being very different from any expected values. This does not indicate an error in the calibration; they simply have no specific true values.

When there are consecutive parallel joint axes in a robot's model, the parameters that define the common normal between the axes are no longer unique. Hence, this issue is another cause of ambiguous parameters in a DH model. If the joint axes are nearly (but not exactly) parallel, then the normal becomes unique. However, this situation causes small errors in the alignment of the axes to produce large changes in the parameters. As a result, many calibration techniques have trouble performing the calibration. To avoid this problem (in some cases), an alternate robot modeling convention based on the Hayati system is often used [134].

Another problem can occur if there are any fixed (immobile) joints in the robot. Some parameters for these joints will be linearly dependent on other parameters (per section 5.3.2.2), which means that they cannot be uniquely identified. This can cause issues in some calibration techniques or simply produce results that are unexpected (but correct).

In summary, one cannot expect there to be a single true DH model for a robot. There may be an infinite number of equally correct models. A model might not match the physical structure of the robot but still produce correct results. No assumption should be made that a calibrated model

must match a given "true" model is invalid. A calibrated model should always be tested using objective metrics like end-effector positional accuracy to determine if the model is correct.

5.5.2 Lack of Orientation Optimization

The fixed-point calibration technique does not utilize any external measurement systems. Instead, it substitutes those measurements with constraints upon the end-effector's pose. Specifically, the end-effector is kept at a fixed point, but its orientation is altered.

This means that no information is available about the orientation of the end-effector. Consequently, the calibration technique cannot directly attempt to improve the orientation accuracy of that is produced by the robot's model. Only the positional accuracy of the end-effector is optimized. It is possible that the orientation accuracy will improve as the parameters are calibrated for positional accuracy, but it is not guaranteed.

If optimization of the end-effector's orientation is required, a different calibration technique can be employed. A device that measures the orientation of the end-effector can be combined with an open-loop calibration technique to improve the orientation accuracy (section 5.3.2). In addition, there are alternative closed-loop techniques that constrain the end-effector's orientation that could be applied (section 5.3.3).

5.5.3 Necessity of Registration

As mentioned under sections 5.3.3 and 5.3.4, there are some limitations to a closed-loop or fixed-point calibration technique. The scale of the robot may be unknown, a final orientation parameter may not be identifiable, and other parameters related to the robot's base may be unidentifiable. These are inherent issues due to the limitations of the collected closed-loop data.

The registration process compensates for the scale and base parameter problems simultaneously. In addition, it enables the robot to operate in some external coordinate system.

In many cases, a registration would be needed to operate in alternate coordinate system regardless of the technique. Therefore, solving many problems via single registration process can be a reasonable decision.

If registration is not desired, there are other ways around the problems. For example, a length parameter of the robot can be independently measured to set the scale. Alternatively, more complex techniques and external measurements systems can help avoid these issues.

5.6 Experiments

5.6.1 Overview

Three different types of experiments were performed using four different robot models. The hypothesis of the experiments was that the proposed calibration technique can improve the positional accuracy of the end-effector for the robot models. The first two experiments were performed entirely in simulation to eliminate variables such as the error inherent in a physical measurement device. The third experiment was performed using two real robots to demonstrate that the proposed technique can be applied in the real world.

The calibration algorithms and simulations were implemented in MATLAB 7.10 (release 2010a, 32-bit). In addition, the Robotics Toolbox (release 8) [135] for MATLAB was employed, which provides many useful functions for robotics. All of the software was executed on a computer with 3 GiB of RAM and a 3.17 GHz Core 2 Duo processor (Intel, Santa Clara, CA) running Windows 7 (Microsoft, Redmond, WA).

5.6.2 Robot Models

5.6.2.1 AESOP 1000

The AESOP, produced by Computer Motion (now Intuitive Surgical), is a medical robot designed to position an endoscope during surgery. Several different versions were made, but the

AESOP 1000 was the one used for the experiments. The AESOP is an excellent choice for this research because it is capable of positioning a device similar to a sensor inside the human body, and it has been used in numerous medical procedures around the world.

The AESOP 1000 has six variable joints (and one fixed joint) connected via links in a serial fashion. Consequently, it supports a full six degrees of freedom at the end-effector. The first joint is prismatic, and the other joints are revolute. The forth joint is a fixed revolute joint that must be locked into one of several positions. It was kept in one position (that extends the arm perfectly horizontally) for all the experiments.

The first three joints and the last joint are actively controlled with motors. The remaining joints are passive, meaning they must be moved by an external force (e.g., by hand). Alternatively, if the tool held by the AESOP 1000 is inserted through an opening that allows rotation (like a laparoscopic port), the end-effector can be actively directed to any position. However, this comes at the expense of a reduced number of degrees of freedom (i.e., no control over the end-effector's yaw or pitch). As described elsewhere [104], a custom motion control system was developed to provide complete control of the AESOP from a PC.

The last joint of the AESOP 1000 controls the roll of the end-effector; it has no effect on the end-effector's position. Since the presented calibration technique only uses position information, the value of the last joint is of no value. This means that from the perspective of the calibration technique, the AESOP 1000 is similar to a robot with one less degree of freedom.

A model of the AESOP 1000 was developed using original DH parameters. The values of the DH parameters are presented in section 5.7.1.1. Note that the axes of joints 1, 2, and 3 are all parallel. In addition, the minimum and maximum joint values (for rotation or translation) were determined for each of the robot's joints. This information is found in Table 5.4.

Joint, *i*	Minimum Value	Maximum Value	Unit
1	0.000	320.000	mm
2	−2.1062	2.1062	rad
3	−2.7118	2.7118	rad
4	0.0000	0.0000	rad
5	−2.3562	2.3562	rad
6	−1.5708	1.5708	rad
7	−3.1416	3.1416	rad

Table 5.4: Measured limits on the joint values of the AESOP 1000.

5.6.2.2 MicroScribe G2X

The MicroScribe G2X, produced by Immersion, is a serial tracking arm with five passive revolute joints. Although it lacks any active joints, it can be positioned manually with ease. One could argue that the lack of active joint control means that it does not meet the proper definition of a robot, but it is sufficient for this research. The MicroScribe G2X has encoders that relay the angles of every joint with a relatively high level of accuracy [93]. This accuracy, as well as its limited number of degrees of freedom (five), makes the MicroScribe G2X an interesting test case for the experiments.

A model of the MicroScribe G2X was developed using Craig's modified DH parameters. The values of the DH parameters are presented in section 5.7.1.2. Note that the axes of joints 2 and 3 are almost exactly parallel. The last "joint" in the model actually represents a fixed transformation that aligns the final coordinate frame with the tip of a stylus that serves as the arm's end-effector.

The minimum and maximum joint angles of the MicroScribe G2X were not measured because active joint manipulation was not an option. For the purposes of simulation, the joints were assumed to be able to rotate a full 2π rad. The actual ranges of the MicroScribe G2X's joints are less than 2π rad, meaning the real device has a somewhat smaller working envelope.

The fact that the developed model for the MicroScribe G2X uses modified DH parameters is significant for the testing. Although original DH convention and Craig's modified DH convention are conceptually very similar, the parameter definitions—and thus the kinematic equations—are different. Despite the differences, no change is made to the presented calibration algorithms other than substituting the forward kinematics equation for modified DH parameters (Equation 2.10). Consequently, the experiments with the MicroScribe G2X should indicate whether the calibration algorithms are flexible enough to support an alternative convention for the robotic model.

5.6.2.3 PUMA 560

The PUMA, produced by Unimation in the late 1970s, is a serial manipulator originally used for industrial applications. The specific model used for this research (entirely in simulation) is the PUMA 560. The PUMA 560 was chosen because it is often used as a test model for robotics research [136]. In addition, the PUMA may have been the first robot used in a surgical procedure (a brain biopsy) [137].

The PUMA 560 has six active revolute joints. The first three joints can be used to control the end-effector's position, and the last three joints form a spherical wrist that can be used to control the orientation. An original DH model of the PUMA 560 was taken from the Robotics Toolbox (release 8) [135] for MATLAB. The values of the DH parameters are presented in section 5.7.1.3. Note that the axes of joints 2 and 3 are parallel.

The minimum and maximum joint angles of the PUMA 560 were not included in the model. Thus, the joints were assumed to be able to rotate a full 2π rad in the simulation. Most of the actual ranges of the PUMA 560's joints are less than 2π rad, meaning the real robot has a somewhat smaller working envelope.

Of special interest is the fact that the last three joints of the model only control the orientation of the end-effector. They have no effect on its position because there is no link or tool after the spherical wrist. Since the presented calibration technique only uses position information, the values of the last three joints are essentially ignored. This means that from the perspective of the calibration technique, the PUMA 560 model is basically equivalent to a robot with only three degrees of freedom. While this may seem insufficient for a technique that requires multiple poses to achieve the same position, there are two factors that make calibration feasible.

First, there are multiple solutions by which the PUMA 560 can achieve a designated pose of the end-effector (assuming it is not at a singularity) [138]. Even ignoring the last three joints, four different configurations of the first three joints can be used to reach a specific position. Second, the proposed calibration technique allows multiple fixed points to be used for the input data. Using multiple fixed points (with four poses per point), sufficient data can be collected for the calibration algorithms.

5.6.2.4 Stanford Arm

The Stanford Arm, designed by Victor Scheinman at Stanford University in the 1960s, is a serial manipulator with six degrees of freedom. It was one of the first computer-controlled robots. Like the PUMA 560, a simulated model of the Stanford Arm was used for this research. The Stanford Arm was selected because it is a well-known robot that is commonly used as a model in the literature on robotics.

The Stanford Arm has six active joints. The third joint is prismatic, while all the other joints are revolute. The last three joints form a spherical wrist. Unlike the model of the PUMA 560, there is a link after the spherical wrist. Therefore, these joints do affect the position of the end-effector, with the exception of the last joint.

The last joint of the Stanford Arm alters the roll of the end-effector without affecting its position. Since the presented calibration technique only uses position information, the value of the last joint is of no value. This means that from the perspective of the calibration technique, the Stanford Arm is similar to a robot with one less degree of freedom.

An original DH model of the Stanford Arm was taken from the Robotics Toolbox (release 8) [135] for MATLAB. The values of the DH parameters are presented in section 5.7.1.4. Note that the axes of joints 3 and 4 are parallel.

The minimum and maximum joint values of the Stanford Arm were not included in the model. Thus, some assumptions were made for the simulation. The revolute joints were allowed to rotate a full 2π rad, and the prismatic joint was given a range of 1 m. The actual ranges of the Stanford Arm's joints are likely less than the assumptions, meaning the real robot probably has a smaller working envelope.

5.6.3 Ideal Simulation Experiment

The ideal simulation experiment was performed using all four robot models in a simulated environment. In this experiment, there was no noise/error added to the feedback from each joint. The only source of error was the modification of each robot's true kinematic parameters.

The ideal simulation experiment was executed to validate the presented calibration technique. Since there was no error added to the measurements, the algorithms were expected to achieve nearly ideal results. Any significant deviation from this expectation could indicate a problem with the technique or its implementation.

5.6.3.1 Generation of Test Joint Configurations

The DH model for each robot from section 5.6.2 was considered to be the true, correct model. For each true robot model, several fixed points within the robot's working envelope were

arbitrarily chosen. Then, for each fixed point, a number of joint configurations that would place the robot's end-effector at the point were calculated (with no concern for the orientation). The joint configurations were computed by feeding arbitrary starting joint values into an iterative inverse kinematics algorithm based on the pseudoinverse of the manipulator Jacobian matrix [135, 139]. Table 5.5 summarizes the numbers of fixed points and joint configurations chosen for each robot.

Robot Model	Chosen Points	Configurations per Point	Total Configurations	Unknown Parameters
AESOP 1000	5	15	75	28
MicroScribe G2X	5	15	75	24
PUMA 560	8	4	32	24
Stanford Arm	6	12	72	24

Table 5.5: Numbers of fixed points and joint configurations chosen for each robot in the simulation. The number of (unknown) DH parameters in each robot model is shown for comparison.

As shown in Table 5.5, the number of total test joint configurations was larger than the number of unknown parameters in each robot's model. In terms of Equation 5.8 (explained in section 5.4.2), this meant that $3n_{\mathrm{jc}} \gg 4n_{\mathrm{j}}$. Thus, there was sufficient data for the calibration algorithms.

The quantity of data for the PUMA 560 was somewhat different from that of the other robots. As explained in section 5.6.2.3, its last three joints have no effect on the end-effector's position, so there are a limited number of effectively unique configurations that can reach a fixed point. Therefore, the quantity of fixed points was increased for the PUMA 560 to compensate.

5.6.3.2 Modification of Robot Models

A modified version of each robot was produced by perturbing the kinematic (DH) parameters of the robot's true model. Each perturbed robot model represented a robot in need of kinematic calibration. To modify the robot models, each non-variable length parameter (a_i, and d_i of non-

prismatic joints) was arbitrarily perturbed by up to ±15 mm, and each non-variable angular parameter (α_i, and θ_i of non-revolute joints) was arbitrarily perturbed by up to ±0.25 rad. In addition, for the AESOP and MicroScribe models, the fixed offset (θ_i^{off} or d_i^{off}) for each variable parameter was perturbed in the same manner. The DH parameters of the perturbed robot models can be found alongside the true and calibrated models under section 5.7.1.

5.6.3.3 Generation of Registration Data

The presented registration algorithm requires a number of known points and their positional coordinates in both a chosen coordinate system and the unregistered robot's base coordinate system. Since the desired outputs of this experiment were robot models and parameters that effectively match the true models as closely as possible, the coordinate system chosen for each robot was the base coordinate system of the robot's true model.

To generate a registration point within the base coordinate system of each robot's true model, each variable joint of the robot was set to a random value from a uniform distribution within the joint's minimum/maximum limits (described under section 5.6.2). Then, using forward kinematics calculations, the positional coordinates of the true model's end-effector were computed. This process was executed seven times to generate seven registration points (and their coordinates in the true model's base coordinate system) for each robot.

Using this method to generate registration points was convenient because it provided joint configurations (the randomly generated values) that would enable the robot to reach each registration point. The positional coordinates of each registration point in the unregistered robot's base coordinate system were calculated by running the same joint configurations through the forward kinematics equations using the unregistered robot model. Thus, the positional

coordinates of each registration point were found in both the chosen coordinate system (the true model's base coordinate system) and the unregistered model's base coordinate system.

5.6.3.4 Execution of Kinematic Calibration/Registration

For each robot, its perturbed model and the generated test joint configurations (grouped by fixed point) were run through the presented calibration algorithms to produce a calibrated model. Next, registration data (the sets of positional coordinates described above) was generated using the true model and the calibrated (but unregistered) model for each robot. This data was then fed into the presented registration algorithm to produce registration parameters for each calibrated robot model. The calibrated model and corresponding registration parameters were the final outputs for each of the four robots.

5.6.3.5 Evaluation of Calibrated/Registered Models

Each of the four calibrated and registered models were evaluated in terms of the errors in the end-effector's position and orientation. The errors were determined by the difference between the true model's end-effector pose and the calibrated/registered model's end-effector pose when the robot was set to a particular joint configuration. Specifically, the position error (e_{p}) was defined as the Euclidean distance between the end-effector position of the true model ($\boldsymbol{p}_{\mathrm{t}}$) and the end-effector position of the calibrated/registered model ($\boldsymbol{p}_{\mathrm{c}}$). This is shown in Equation 5.19.

$$e_{\mathrm{p}} = \|\boldsymbol{p}_{\mathrm{t}} - \boldsymbol{p}_{\mathrm{c}}\| \qquad \text{Equation 5.19}$$

The orientation error (e_{o}) was defined as the (shortest) angle between the end-effector orientation of the true model and the end-effector orientation of the calibrated/registered model. This was computed according to Equation 5.20, where $\boldsymbol{q}_{\mathrm{t}}$ and $\boldsymbol{q}_{\mathrm{c}}$ are the unit quaternion representations of the true and calibrated/registered orientations (respectively) and the "$\cdot$"

operator represents the dot product. The equations needed to convert a 3×3 rotation matrix to a unit quaternion can be found in [111].

$$e_{\mathrm{o}} = 2\arccos|\boldsymbol{q}_{\mathrm{t}} \cdot \boldsymbol{q}_{\mathrm{c}}| \qquad \text{Equation 5.20}$$

There are an infinite number of possible joint configurations for each robot, and each configuration can lead a different amount of position error and orientation error. To provide an overall representation of the position and orientation error of each model, the mean position error and mean orientation error were computed using 1000 joint configurations per robot. In addition, the 95% confidence interval around each mean was calculated to determine the reliability of the estimated (mean) error.

To generate one of the 1000 joints configurations per robot, each variable joint of the robot was set to a random value from a uniform distribution within the joint's minimum/maximum limits (described under section 5.6.2). This was performed 1000 times to produce 1000 random joint configurations. The forward kinematics equations (section 2.3.1.2) were then used to determine the end-effector pose for each model when using each joint configuration.

To quantify any improvements provided by the calibration/registration, the same error calculations were performed using the same joint configurations for each of the four (initial) perturbed models. Thus, the mean position and orientation errors of each robot before the calibration/registration were also obtained. By comparing the errors before and after calibration/registration, the effects of the calibration/registration can be understood.

5.6.4 Noisy Simulation Experiment

The noisy simulation experiment was performed using all four robot models in a simulated environment. In this experiment, there was noise added to the feedback from each variable joint.

In addition, each robot's true kinematic parameters were modified to introduce error. The noise was introduced to challenge the technique while remaining in the controlled environment of a simulation.

5.6.4.1 Generation of Test Joint Configurations

The same test joint configurations as those described in section 5.6.3.1 (for the ideal simulation experiment) were used as a starting point. Next, random values were added to the joint values in the test joint configurations to simulate noise/error in the joint feedback. A random value between ±1 mrad (taken from a uniform distribution) was added to each revolute joint angle in each test joint configuration. Similarly, a random value between ±1 mm (taken from a uniform distribution) was added to each prismatic joint value in each test joint configuration.

Even though real encoders can be more accurate than implied by the chosen random values, there are many other potential sources of error in a robotic system. Some examples include backlash, joint/link compliance, and inexact end-effector placement (or measurement). The added random values were merely a simplification of the sources of error in a real robot.

5.6.4.2 Modification of Robot Models

The same perturbed robots models as those used for the ideal simulation experiment were used for the noisy simulation experiment. The DH parameters of the perturbed robot models can be found alongside the true and calibrated models under section 5.7.1.

5.6.4.3 Generation of Registration Data

The same registration points as those described in section 5.6.3.3 (for the ideal simulation experiment) were used as a starting point. Therefore, the positional coordinates of the

registration points in the chosen coordinate system (the base coordinate system of each robot's true model) were the same. However, for the unregistered models, random values were added to the joint values in the registration joint configurations to simulate noise/error in the joint feedback. Since these joint configurations were used to calculate (via forward kinematics) the positional coordinates of each registration point in the unregistered robot's base coordinate system, these positional coordinates were perturbed.

The random values added to the joint values were taken from the same distributions as those used for the noisy test joint configurations (from section 5.6.4.1). Thus, each joint value was modified by an amount between ±1 mrad (for revolute joints) or ±1 mm (for prismatic joints).

5.6.4.4 Execution of Kinematic Calibration/Registration

The same basic process was used to execute the noisy kinematic calibration/registration as the one used for the ideal kinematic calibration/registration (described in section 5.6.3.4). The only differences were in the data that was used. For the calibration algorithms, the perturbed robot models were used along with the randomly modified test joint configurations (grouped by fixed point). For the registration algorithm, the ideal positional coordinates of the registration points in each true model's base coordinate system were used along with the perturbed positional coordinates of the registration points in each calibrated (but unregistered) model's base coordinate system. The calibrated model and corresponding registration parameters were the final outputs for each of the four robots.

5.6.4.5 Evaluation of Calibrated/Registered Models

The same process was used to evaluate the noisy calibrated/registered models as the one used for the ideal calibrated/registered models (described in section 5.6.3.5). The same 1000 joint configurations per robot were used to estimate the mean position error and mean orientation error

of each calibrated/registered model. In addition, the same error results for the (initial) perturbed models can be used to highlight any improvements provided by the calibration/registration.

5.6.5 Real Robot Experiment

The real robot experiment was performed using two actual robots (the AESOP 1000 and the MicroScribe G2X) in the real world. In this experiment, actual joint measurements from multiple fixed points were used to calibrate the robots' models. More specifically, leave-one-out calibrations were performed to avoid biasing the results.

5.6.5.1 Acquisition of AESOP 1000 Test Joint Configurations

The AESOP 1000 has encoders and potentiometers that it uses to acquire feedback from its variable joints. Unfortunately, this feedback is not directly available to external systems, so a custom motion control system had to be developed. The details of the motion control system are provided in [104]. In summary, the system enables all the active joints of the AESOP to be controlled, and it provides the values of all joints in real time.

Four fixed points on flat surfaces were arbitrarily chosen within the AESOP's working envelope. Using a combination of active joint control and manual manipulation of the passive joints, the AESOP's end-effector was placed into numerous poses that reached the established fixed points. The poses were chosen somewhat arbitrarily, but an effort was made to move each joint through a broad range and approach the fixed points from diverse orientations. After the AESOP was placed in a pose that reached a fixed point, the current joint configuration was recorded.

In the end, a total of 79 joint configurations were acquired. There were 20, 20, 20, and 19 poses recorded for each fixed point, respectively. As shown in Table 5.5 in section 5.6.3.1, the

AESOP 1000 only has 28 unknown DH parameters. Thus, there was sufficient data for the calibration algorithms.

5.6.5.2 Acquisition of MicroScribe G2X Test Joint Configurations

The MicroScribe G2X has optical encoders that measure the angles of its five variable joints. The device comes with software that allows all the joint angles to be viewed. Thus, it was easy to record joint configurations.

Five fixed points on flat surfaces were arbitrarily chosen within the MicroScribe's working envelope. Since the MicroScribe G2X is a passive tracking device, it was manually manipulated to place its end-effector into numerous poses that reached the established fixed points. The poses were chosen somewhat arbitrarily, but an effort was made to move each joint through a broad range and approach the fixed points from diverse orientations. After the MicroScribe was placed in a pose that reached a fixed point, the current joint configuration was recorded.

In the end, a total of 90 joint configurations were acquired. There were 18, 17, 19, 14, and 22 poses recorded for each fixed point, respectively. As shown in Table 5.5 in section 5.6.3.1, the MicroScribe G2X only has 24 unknown DH parameters. Thus, there was sufficient data for the calibration algorithms.

5.6.5.3 Determination of Initial Robot Models

There was no DH model of the AESOP 1000 available from the manufacturer. Thus, the initial model was generated from manual measurements of the robot's links and joints (e.g., using a caliper). The process that was followed is described in [104]. Since the measurements were made by hand on an assembled robot, it was expected that there would be some error in the model. In addition, usage of the robot subsequent to the generation of the model caused wear that may have added further error to the initial DH model.

For the MicroScribe G2X, a relatively accurate DH model was available. This was used as the initial model for the calibration. Any error in the model was likely caused by limitations in the manufacturing/assembly process or wear to the device.

5.6.5.4 Selection of an Error Metric

Unlike the simulation experiments, there was no known true model of either robot for the real robot experiment. In addition, there was no registration data available. Therefore, a different approach was required for the evaluation (and execution) of the kinematic calibration.

The presented calibration technique presumes that all poses used to reach a fixed point have the end-effector's tip at the same position. Assuming the end-effector is placed correctly, any deviation of the end-effector's calculated position (obtained via forward kinematics) from the location of the fixed point is due to error in the system. Although there are a number of potential sources of error in a real robotic system, errors in the DH model will definitely cause deviation of the end-effector's calculated position from the location of the fixed point. In general, a less accurate DH model will cause more deviation, and a more accurate DH model will result in less deviation. Thus, the deviations of the end-effector's calculated positions from the fixed points can be used as a measure of the DH model's relative accuracy (section 5.4.4.2).

The exact locations of the fixed points relative to the robot's base coordinate system were not known. They could have been measured with an external measurement device, but even then there would have been error in the external measurements. Instead, the location of each fixed point was estimated to be the mean of the end-effector positions (calculated via forward kinematics) of all the poses that were used to reach the point. Section 5.4.4.2 contains the relevant calculations. Even if the estimated fixed-point locations weren't absolutely correct, they should still function similar to the true fixed-point locations for the accuracy assessment. In

general, a less precise DH model will have more deviation from the estimated fixed points (a wider spread), and a more precise DH model will have less deviation (a tighter spread).

No data was collected that could be used to register either robot. If there had been, the presented registration process could have been used to set the scale of the robot and adjust the location and orientation of the robot's base in an external coordinate system. However, since it was desired to assess the internal accuracy of the DH models, the location and orientation of the robot's base could be neglected. The same deviations would be found when the fixed point's location was expressed in the robot's local coordinate system instead of a translated/rotated external coordinate system.

On the other hand, the scale of a robot would have an effect on the calculated deviations. Scaling a robot by a certain factor would result in the computed deviations being scaled by the same factor. Thus, it would have been beneficial to collect registration data to help set the scale (particularly for the MicroScribe, which lacks a prismatic joint). However, results from simulations showed that the scale typically drifts by no more than a few percent during calibration. In addition, the implemented calibration algorithms limit how much the DH parameters can change, thus limiting the change in scale. Therefore, it was deemed reasonable to proceed without registration.

In summary, the mean of the deviations of the end-effector's calculated position from each corresponding estimated fixed-point location was chosen as the metric for comparing each calibrated DH model to the initial model. Although this was not as complete as the evaluation used in the simulation experiments, it provided a seemingly reasonable estimate of the accuracy. As explained below, multiple models were generated for each robot, which were used to demonstrate the consistency of the estimated accuracy.

5.6.5.5 Execution of Kinematic Calibration

Unlike the simulation experiments, the true models were not known in the real robot experiment, and there were practical limits to the number of joint configurations that could be measured. For example, it would not be possible to generate 1000 new joint configurations to evaluate each model. Instead, the data used to evaluate the system would have to be drawn from the limited set of measurements that were made.

To avoid biasing the results, the data used to evaluate each model should be separate from the data used to calibrate the model. This posed a problem because limited data was collected, and the calibration should be more accurate when performed with more data (at least to a certain extent). Leave-one-out calibrations were used to mitigate this problem.

For the leave-one-out calibrations of each robot, the recorded joint configurations were grouped according the fixed point they reached. Next, a calibration was performed using the data from all of the groups except one. The excluded group would later be used to evaluate the calibrated model. This process was repeated until every group had the opportunity to be the excluded one.

There were four fixed points used for the AESOP 1000 and five fixed points used for the MicroScribe G2X. Consequently, there were four sets of joint configurations employed in the AESOP calibration (along with the common initial DH model), and there were five sets of joint configurations employed in the MicroScribe calibration (along with the common initial model). Thus, the outputs of the kinematic calibrations were four different calibrated models of the AESOP 1000 and five different calibrated models of the MicroScribe G2X. Note that no registrations were performed, meaning no registration parameters were produced.

5.6.5.6 Evaluation of Calibrated Models

As explained in section 5.6.5.4, the metric selected to evaluate each robot's models was the mean of the deviations of the end-effector's calculated position from each corresponding estimated fixed-point location (per section 5.4.4.2). To convey exactly what this entailed, the evaluation of the first calibrated model for the AESOP 1000 will be described in detail. The evaluations of the other calibrated models followed the same procedure, each using a different excluded fixed point for the calculations.

The first calibrated model of the AESOP 1000 was previously generated using the 59 measured joint configurations that reached all the fixed points except the first one. Therefore, the 20 joint configurations that reached the first fixed point were used for this evaluation. To estimate the location of the first fixed point, the joint configurations that reached the first point were run through the forward kinematics equations (section 2.3.1.2) using the calibrated model. The outputs were 20 different end-effector positions that were intended to be at the first fixed point. The mean of these 20 positions was used as the estimated location of the first fixed point.

Next, the Euclidean distance between each of the 20 calculated end-effector positions and the estimated first fixed-point location was computed. Finally, the average of these 20 distances was calculated to produce the mean deviation of the end-effector's calculated position from the estimated first fixed-point location. This represented the desired error metric.

In addition to being applied to the other calibrated models, the same evaluation process was followed while using each robot's initial model in place of the calibrated models. In other words, the four mean deviations were calculated for the initial AESOP 1000 model, and the five mean deviations were calculated for the initial MicroScribe G2X model. This was done so the errors before the calibrations could be compared to the errors after the calibrations.

5.7 Results

The results from each of three experiments (ideal simulation, noisy simulation, and real robot) are presented in the sections below. As appropriate, execution times, calibrated models, registration parameters, and calculated errors are presented.

5.7.1 Ideal Simulation Results

The results of the ideal simulation experiment (described under section 5.6.3) are shown below. The results are grouped according to each of the robots used in the experiment. The average execution time of the calibration algorithms for all four robots was 56.4 minutes on the test computer (described in section 5.6.1).

5.7.1.1 AESOP 1000

The calibrated model of the AESOP 1000 that was derived from the ideal simulation is presented in Table 5.6 (using the original DH convention). In addition, the true model and the perturbed model (upon which the calibration was run) are included for comparison. The terms with the superscript "var" represent variable joint values. The parameters that are shaded in gray were found to be "ineffective"; they could not be calibrated with the data used by the fixed-point calibration technique.

	Joint, i	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
AESOP 1000 True Model	1	0.0000	0.000	0.0000	$d_1^{\text{var}} + 0.000$
	2	0.0000	384.440	$\theta_2^{\text{var}} + 0.0000$	0.000
	3	1.5708	52.710	$\theta_3^{\text{var}} + 0.0000$	0.000
	4	1.5708	0.000	1.5708	0.000
	5	1.5708	0.000	$\theta_5^{\text{var}} + 3.1416$	250.700
	6	−1.5708	17.200	$\theta_6^{\text{var}} + 1.5708$	0.000
	7	0.0000	0.000	$\theta_7^{\text{var}} - 1.5708$	−275.420
AESOP 1000 Perturbed Model	**Joint, i**	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
	1	0.1000	5.000	−0.2000	$d_1^{\text{var}} - 10.000$
	2	0.2000	399.440	$\theta_2^{\text{var}} - 0.2000$	−5.000
	3	1.3208	62.710	$\theta_3^{\text{var}} + 0.2000$	−5.000
	4	1.3708	−15.000	1.6708	10.000
	5	1.6708	5.000	$\theta_5^{\text{var}} + 3.0416$	260.700
	6	−1.7708	7.200	$\theta_6^{\text{var}} + 1.8208$	−10.000
	7	0.2000	−5.000	$\theta_7^{\text{var}} - 1.7708$	−260.420
AESOP 1000 Calibrated Model (Ideal)	**Joint, i**	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
	1	0.0000	5.000	−0.2000	$d_1^{\text{var}} - 10.000$
	2	0.0000	384.440	$\theta_2^{\text{var}} - 0.2000$	−5.000
	3	1.4632	50.345	$\theta_3^{\text{var}} + 0.4116$	−5.000
	4	1.1617	−34.883	1.5240	17.628
	5	1.5708	0.000	$\theta_5^{\text{var}} + 3.2589$	250.258
	6	−1.5909	5.714	$\theta_6^{\text{var}} + 1.6125$	−5.559
	7	0.2000	0.000	$\theta_7^{\text{var}} - 1.7708$	−275.953

Table 5.6: Denavit–Hartenberg parameters of the AESOP 1000 for its true model, the perturbed model, and the calibrated model generated from the ideal simulation. The parameters shaded in gray could not be calibrated.

The registration parameters (for translation, rotation, and scaling of the calibrated model) computed by the registration algorithm (using ideal joint values) are found in Table 5.7. The values at the bottom indicate the overall changes in robot position and orientation effected by the parameters.

Translation			**Rotation**			**Scale**
Δx (mm)	Δy (mm)	Δz (mm)	φ (rad)	θ (rad)	ψ (rad)	s
−4.900	−0.993	52.750	−2.6249	0.0000	3.0249	1.0000
Change in Position			**Change in Orientation**			
52.987 mm			0.4000 rad			

Table 5.7: Registration parameters and their overall effects from the ideal post-calibration registration of the AESOP 1000.

The mean position error and mean orientation error of the end-effector both before and after calibration/registration are displayed in Table 5.8. The errors were derived from the simulation of 1000 random joint configurations. Using this information, the 95% confidence interval around each mean was calculated to indicate the reliability of the estimated (mean) error.

Model	**Mean Position Error** (mm)	**Mean Orientation Error** (rad)
AESOP Perturbed	277.139 ± 6.279	0.8125 ± 0.0136
AESOP Calibrated	0.000 ± 0.000	0.2863 ± 0.0014

Table 5.8: Comparison of the mean errors of the end-effector's position and orientation before and after ideal calibration/registration of the AESOP 1000. The 95% confidence interval for each mean is provided.

5.7.1.2 MicroScribe G2X

The calibrated model of the MicroScribe G2X that was derived from the ideal simulation is presented in Table 5.9 (using Craig's modified DH convention). In addition, the true model and the perturbed model (upon which the calibration was run) are included for comparison. The terms with the superscript "var" represent variable joint angles. The parameters that are shaded in gray were found to be "ineffective"; they could not be calibrated with the data used by the fixed-point calibration technique.

	Joint, i	α_{i-1} (rad)	a_{i-1} (mm)	θ_i (rad)	d_i (mm)
MicroScribe G2X True Model	1	0.0000	0.000	$\theta_1^{\text{var}} + 0.0000$	210.820
	2	1.5705	24.181	$\theta_2^{\text{var}} + 0.0000$	–22.530
	3	–0.0006	260.680	$\theta_3^{\text{var}} + 0.0000$	–0.305
	4	1.5621	13.894	$\theta_4^{\text{var}} + 0.0000$	234.696
	5	–1.5731	–10.262	$\theta_5^{\text{var}} + 0.0000$	8.103
	6	–1.5690	10.160	0.0000	–134.163
	Joint, i	α_{i-1} (rad)	a_{i-1} (mm)	θ_i (rad)	d_i (mm)
MicroScribe G2X Perturbed Model	1	0.1000	5.000	$\theta_1^{\text{var}} - 0.2000$	200.820
	2	1.3705	14.181	$\theta_2^{\text{var}} + 0.2000$	–7.530
	3	–0.1006	275.680	$\theta_3^{\text{var}} - 0.1000$	–5.305
	4	1.3121	23.894	$\theta_4^{\text{var}} + 0.2500$	219.696
	5	–1.3731	–15.262	$\theta_5^{\text{var}} - 0.2000$	13.103
	6	–1.3190	0.160	−0.2000	–124.163
	Joint, i	α_{i-1} (rad)	a_{i-1} (mm)	θ_i (rad)	d_i (mm)
MicroScribe G2X Calibrated Model (Ideal)	1	0.1000	5.000	$\theta_1^{\text{var}} - 0.2000$	200.820
	2	1.5705	23.956	$\theta_2^{\text{var}} + 0.0000$	–0.912
	3	–0.0006	258.260	$\theta_3^{\text{var}} + 0.0000$	–21.710
	4	1.5621	13.765	$\theta_4^{\text{var}} + 0.0000$	232.517
	5	–1.5731	–10.166	$\theta_5^{\text{var}} + 0.0754$	32.955
	6	–1.3842	0.030	−0.2000	–135.653

Table 5.9: Denavit–Hartenberg parameters of the MicroScribe G2X for its true model, the perturbed model, and the calibrated model generated from the ideal simulation. The parameters shaded in gray could not be calibrated.

The registration parameters (for translation, rotation, and scaling of the calibrated model) computed by the registration algorithm (using ideal joint values) are found in Table 5.10. The values at the bottom indicate the overall changes in robot position and orientation effected by the parameters.

Translation			**Rotation**			**Scale**
Δx (mm)	Δy (mm)	Δz (mm)	φ (rad)	θ (rad)	ψ (rad)	s
−4.946	−1.003	8.118	1.7708	0.1000	−1.5708	1.0094
Change in Position			**Change in Orientation**			
9.559 mm			0.2235 rad			

Table 5.10: Registration parameters and their overall effects from the ideal post-calibration registration of the MicroScribe G2X.

The mean position error and mean orientation error of the end-effector both before and after calibration/registration are displayed in Table 5.11. The errors were derived from the simulation of 1000 random joint configurations. Using this information, the 95% confidence interval around each mean was calculated to indicate the reliability of the estimated (mean) error.

Model	**Mean Position Error** (mm)	**Mean Orientation Error** (rad)
MicroScribe Perturbed	127.732 ± 3.723	0.5998 ± 0.0143
MicroScribe Calibrated	0.000 ± 0.000	0.2772 ± 0.0000

Table 5.11: Comparison of the mean errors of the end-effector's position and orientation before and after ideal calibration/registration of the MicroScribe G2X. The 95% confidence interval for each mean is provided.

5.7.1.3 PUMA 560

The calibrated model of the PUMA 560 that was derived from the ideal simulation is presented in Table 5.12 (using the original DH convention). In addition, the true model and the perturbed model (upon which the calibration was run) are included for comparison. The terms with the superscript "var" represent variable joint angles. The parameters that are shaded in gray were found to be "ineffective"; they could not be calibrated with the data used by the fixed-point calibration technique.

	Joint, i	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
PUMA 560 True Model	1	1.5708	0.000	$\theta_1^{\text{var}} + 0.0000$	0.000
	2	0.0000	431.800	$\theta_2^{\text{var}} + 0.0000$	0.000
	3	–1.5708	20.300	$\theta_3^{\text{var}} + 0.0000$	150.050
	4	1.5708	0.000	$\theta_4^{\text{var}} + 0.0000$	431.800
	5	–1.5708	0.000	$\theta_5^{\text{var}} + 0.0000$	0.000
	6	0.0000	0.000	$\theta_6^{\text{var}} + 0.0000$	0.000
	Joint, i	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
PUMA 560 Perturbed Model	1	1.6708	5.000	$\theta_1^{\text{var}} + 0.0000$	–10.000
	2	0.2000	446.800	$\theta_2^{\text{var}} + 0.0000$	–5.000
	3	–1.8208	30.300	$\theta_3^{\text{var}} + 0.0000$	145.050
	4	1.3708	–15.000	$\theta_4^{\text{var}} + 0.0000$	441.800
	5	–1.4708	5.000	$\theta_5^{\text{var}} + 0.0000$	10.000
	6	–0.2000	–10.000	$\theta_6^{\text{var}} + 0.0000$	–10.000
	Joint, i	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
PUMA 560 Calibrated Model (Ideal)	1	1.5708	0.000	$\theta_1^{\text{var}} + 0.0000$	–10.000
	2	0.0000	437.140	$\theta_2^{\text{var}} + 0.0000$	–3.525
	3	–1.5315	30.750	$\theta_3^{\text{var}} + 0.0233$	138.279
	4	1.3708	0.000	$\theta_4^{\text{var}} + 0.0000$	436.878
	5	–1.4708	0.000	$\theta_5^{\text{var}} + 0.0000$	0.000
	6	–0.2000	0.000	$\theta_6^{\text{var}} + 0.0000$	0.000

Table 5.12: Denavit–Hartenberg parameters of the PUMA 560 for its true model, the perturbed model, and the calibrated model generated from the ideal simulation. The parameters shaded in gray could not be calibrated.

The registration parameters (for translation, rotation, and scaling of the calibrated model) computed by the registration algorithm (using ideal joint values) are found in Table 5.13. The values at the bottom indicate the overall changes in robot position and orientation effected by the parameters.

Translation			**Rotation**			**Scale**
Δx (mm)	Δy (mm)	Δz (mm)	φ (rad)	θ (rad)	ψ (rad)	s
0.000	0.000	9.878	0.0000	0.0000	0.0000	0.9878
Change in Position			**Change in Orientation**			
9.878 mm			0.0000 rad			

Table 5.13: Registration parameters and their overall effects from the ideal post-calibration registration of the PUMA 560.

The mean position error and mean orientation error of the end-effector both before and after calibration/registration are displayed in Table 5.14. The errors were derived from the simulation of 1000 random joint configurations. Using this information, the 95% confidence interval around each mean was calculated to indicate the reliability of the estimated (mean) error.

Model	**Mean Position Error** (mm)	**Mean Orientation Error** (rad)
PUMA Perturbed	129.312 ± 4.047	0.4150 ± 0.0106
PUMA Calibrated	0.000 ± 0.000	0.2860 ± 0.0058

Table 5.14: Comparison of the mean errors of the end-effector's position and orientation before and after ideal calibration/registration of the PUMA 560. The 95% confidence interval for each mean is provided.

5.7.1.4 Stanford Arm

The calibrated model of the Stanford Arm that was derived from the ideal simulation is presented in Table 5.15 (using the original DH convention). In addition, the true model and the perturbed model (upon which the calibration was run) are included for comparison. The terms with the superscript "var" represent variable joint values. The parameters that are shaded in gray were found to be "ineffective"; they could not be calibrated with the data used by the fixed-point calibration technique.

	Joint, i	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
Stanford Arm True Model	1	–1.5708	0.000	$\theta_1^{\text{var}} + 0.0000$	412.000
	2	1.5708	0.000	$\theta_2^{\text{var}} + 0.0000$	154.000
	3	0.0000	0.000	–1.5708	$d_3^{\text{var}} + 0.000$
	4	–1.5708	0.000	$\theta_4^{\text{var}} + 0.0000$	0.000
	5	1.5708	0.000	$\theta_5^{\text{var}} + 0.0000$	0.000
	6	0.0000	0.000	$\theta_6^{\text{var}} + 0.0000$	263.000
	Joint, i	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
Stanford Arm Perturbed Model	1	–1.4708	5.000	$\theta_1^{\text{var}} + 0.0000$	402.000
	2	1.7708	15.000	$\theta_2^{\text{var}} + 0.0000$	149.000
	3	–0.2500	10.000	–1.6708	$d_3^{\text{var}} + 0.000$
	4	–1.7708	–15.000	$\theta_4^{\text{var}} + 0.0000$	10.000
	5	1.6708	5.000	$\theta_5^{\text{var}} + 0.0000$	10.000
	6	–0.2000	–10.000	$\theta_6^{\text{var}} + 0.0000$	253.000
	Joint, i	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
Stanford Arm Calibrated Model (Ideal)	1	–1.5708	0.000	$\theta_1^{\text{var}} + 0.0000$	402.000
	2	1.5708	2.427	$\theta_2^{\text{var}} + 0.0000$	168.286
	3	0.0000	14.491	–1.7391	$d_3^{\text{var}} - 7.726$
	4	–1.5708	0.000	$\theta_4^{\text{var}} + 0.1683$	7.726
	5	1.5716	2.708	$\theta_5^{\text{var}} - 0.0103$	0.220
	6	–0.2000	0.000	$\theta_6^{\text{var}} + 0.0000$	262.986

Table 5.15: Denavit–Hartenberg parameters of the Stanford Arm for its true model, the perturbed model, and the calibrated model generated from the ideal simulation. The parameters shaded in gray could not be calibrated.

The registration parameters (for translation, rotation, and scaling of the calibrated model) computed by the registration algorithm (using ideal joint values) are found in Table 5.16. The values at the bottom indicate the overall changes in robot position and orientation effected by the parameters.

Translation			**Rotation**			**Scale**
Δx (mm)	Δy (mm)	Δz (mm)	φ (rad)	θ (rad)	ψ (rad)	s
0.000	0.000	10.000	0.0000	0.0000	0.0000	1.0000
Change in Position			**Change in Orientation**			
10.000 mm			0.0000 rad			

Table 5.16: Registration parameters and their overall effects from the ideal post-calibration registration of the Stanford Arm.

The mean position error and mean orientation error of the end-effector both before and after calibration/registration are displayed in Table 5.17. The errors were derived from the simulation of 1000 random joint configurations. Using this information, the 95% confidence interval around each mean was calculated to indicate the reliability of the estimated (mean) error.

Model	**Mean Position Error** (mm)	**Mean Orientation Error** (rad)
Stanford Arm Perturbed	147.667 ± 4.534	0.4259 ± 0.0104
Stanford Arm Calibrated	0.000 ± 0.000	0.2004 ± 0.0005

Table 5.17: Comparison of the mean errors of the end-effector's position and orientation before and after ideal calibration/registration of the Stanford Arm. The 95% confidence interval for each mean is provided.

5.7.2 Noisy Simulation Results

The results of the noisy simulation experiment (described under section 5.6.4) are presented below. The results are grouped according to each of the robots used in the experiment. The average execution time of the calibration algorithms for all four robots was 34.4 minutes on the test computer (described in section 5.6.1).

5.7.2.1 AESOP 1000

The calibrated model of the AESOP 1000 that was derived from the noisy simulation is presented in Table 5.18 (using the original DH convention). In addition, the true model and the perturbed model (upon which the calibration was run) are included for comparison. The terms with the superscript "var" represent variable joint values. The parameters that are shaded in gray were found to be "ineffective"; they could not be calibrated with the data used by the fixed-point calibration technique.

	Joint, i	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
AESOP 1000 True Model	1	0.0000	0.000	0.0000	$d_1^{\text{var}} + 0.000$
	2	0.0000	384.440	$\theta_2^{\text{var}} + 0.0000$	0.000
	3	1.5708	52.710	$\theta_3^{\text{var}} + 0.0000$	0.000
	4	1.5708	0.000	1.5708	0.000
	5	1.5708	0.000	$\theta_5^{\text{var}} + 3.1416$	250.700
	6	−1.5708	17.200	$\theta_6^{\text{var}} + 1.5708$	0.000
	7	0.0000	0.000	$\theta_7^{\text{var}} - 1.5708$	−275.420
AESOP 1000 Perturbed Model	**Joint, i**	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
	1	0.1000	5.000	−0.2000	$d_1^{\text{var}} - 10.000$
	2	0.2000	399.440	$\theta_2^{\text{var}} - 0.2000$	−5.000
	3	1.3208	62.710	$\theta_3^{\text{var}} + 0.2000$	−5.000
	4	1.3708	−15.000	1.6708	10.000
	5	1.6708	5.000	$\theta_5^{\text{var}} + 3.0416$	260.700
	6	−1.7708	7.200	$\theta_6^{\text{var}} + 1.8208$	−10.000
	7	0.2000	−5.000	$\theta_7^{\text{var}} - 1.7708$	−260.420
AESOP 1000 Calibrated Model (Noisy)	**Joint, i**	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
	1	0.0006	5.000	−0.2000	$d_1^{\text{var}} - 10.000$
	2	0.0004	384.602	$\theta_2^{\text{var}} - 0.5000$	−5.000
	3	1.6105	51.257	$\theta_3^{\text{var}} + 0.3293$	−7.722
	4	1.2428	−34.035	1.5850	19.122
	5	1.5742	0.197	$\theta_5^{\text{var}} + 3.1018$	249.378
	6	−1.6206	9.682	$\theta_6^{\text{var}} + 1.5955$	−12.876
	7	0.2000	−0.113	$\theta_7^{\text{var}} - 1.5251$	−276.382

Table 5.18: Denavit–Hartenberg parameters of the AESOP 1000 for its true model, the perturbed model, and the calibrated model generated from the noisy simulation. The parameters shaded in gray could not be calibrated.

The registration parameters (for translation, rotation, and scaling of the calibrated model) computed by the registration algorithm (using noisy joint values) are found in Table 5.19. The values at the bottom indicate the overall changes in robot position and orientation effected by the parameters.

Translation			**Rotation**			**Scale**
Δx (mm)	Δy (mm)	Δz (mm)	φ (rad)	θ (rad)	ψ (rad)	s
−4.013	−2.893	57.272	2.2852	0.0024	−1.5857	0.9991
Change in Position			**Change in Orientation**			
57.486 mm			0.6995 rad			

Table 5.19: Registration parameters and their overall effects from the noisy post-calibration registration of the AESOP 1000.

The mean position error and mean orientation error of the end-effector both before and after calibration/registration are displayed in Table 5.20. The errors were derived from the simulation of 1000 random joint configurations. Using this information, the 95% confidence interval around each mean was calculated to indicate the reliability of the estimated (mean) error.

Model	**Mean Position Error** (mm)	**Mean Orientation Error** (rad)
AESOP Perturbed	277.139 ± 6.279	0.8125 ± 0.0136
AESOP Calibrated	1.202 ± 0.040	0.2077 ± 0.0023

Table 5.20: Comparison of the mean errors of the end-effector's position and orientation before and after noisy calibration/registration of the AESOP 1000. The 95% confidence interval for each mean is provided.

5.7.2.2 MicroScribe G2X

The calibrated model of the MicroScribe G2X that was derived from the noisy simulation is presented in Table 5.21 (using Craig's modified DH convention). In addition, the true model and the perturbed model (upon which the calibration was run) are included for comparison. The terms with the superscript "var" represent variable joint angles. The parameters that are shaded in gray were found to be "ineffective"; they could not be calibrated with the data used by the fixed-point calibration technique.

	Joint, i	α_{i-1} (rad)	a_{i-1} (mm)	θ_i (rad)	d_i (mm)
MicroScribe G2X True Model	1	0.0000	0.000	$\theta_1^{\text{var}} + 0.0000$	210.820
	2	1.5705	24.181	$\theta_2^{\text{var}} + 0.0000$	–22.530
	3	–0.0006	260.680	$\theta_3^{\text{var}} + 0.0000$	–0.305
	4	1.5621	13.894	$\theta_4^{\text{var}} + 0.0000$	234.696
	5	–1.5731	–10.262	$\theta_5^{\text{var}} + 0.0000$	8.103
	6	–1.5690	10.160	0.0000	–134.163
	Joint, i	**α_{i-1} (rad)**	**a_{i-1} (mm)**	**θ_i (rad)**	**d_i (mm)**
MicroScribe G2X Perturbed Model	1	0.1000	5.000	$\theta_1^{\text{var}} - 0.2000$	200.820
	2	1.3705	14.181	$\theta_2^{\text{var}} + 0.2000$	–7.530
	3	–0.1006	275.680	$\theta_3^{\text{var}} - 0.1000$	–5.305
	4	1.3121	23.894	$\theta_4^{\text{var}} + 0.2500$	219.696
	5	–1.3731	–15.262	$\theta_5^{\text{var}} - 0.2000$	13.103
	6	–1.3190	0.160	−0.2000	–124.163
	Joint, i	**α_{i-1} (rad)**	**a_{i-1} (mm)**	**θ_i (rad)**	**d_i (mm)**
MicroScribe G2X Calibrated Model (Noisy)	1	0.1000	5.000	$\theta_1^{\text{var}} - 0.2000$	200.820
	2	1.5706	23.746	$\theta_2^{\text{var}} + 0.0000$	–10.472
	3	–0.0007	255.680	$\theta_3^{\text{var}} + 0.0002$	–11.913
	4	1.5623	13.600	$\theta_4^{\text{var}} + 0.0000$	230.188
	5	–1.5733	–10.064	$\theta_5^{\text{var}} + 0.0749$	30.590
	6	–1.3992	0.085	−0.2000	–133.986

Table 5.21: Denavit–Hartenberg parameters of the MicroScribe G2X for its true model, the perturbed model, and the calibrated model generated from the noisy simulation. The parameters shaded in gray could not be calibrated.

The registration parameters (for translation, rotation, and scaling of the calibrated model) computed by the registration algorithm (using noisy joint values) are found in Table 5.22. The values at the bottom indicate the overall changes in robot position and orientation effected by the parameters.

Translation			**Rotation**			**Scale**
Δx (mm)	Δy (mm)	Δz (mm)	φ (rad)	θ (rad)	ψ (rad)	s
−4.851	−0.923	6.030	1.7741	0.1004	−1.5740	1.0200
Change in Position			**Change in Orientation**			
7.793 mm			0.2239 rad			

Table 5.22: Registration parameters and their overall effects from the noisy post-calibration registration of the MicroScribe G2X.

The mean position error and mean orientation error of the end-effector both before and after calibration/registration are displayed in Table 5.23. The errors were derived from the simulation of 1000 random joint configurations. Using this information, the 95% confidence interval around each mean was calculated to indicate the reliability of the estimated (mean) error.

Model	**Mean Position Error** (mm)	**Mean Orientation Error** (rad)
MicroScribe Perturbed	127.732 ± 3.723	0.5998 ± 0.0143
MicroScribe Calibrated	0.308 ± 0.007	0.2678 ± 0.0000

Table 5.23: Comparison of the mean errors of the end-effector's position and orientation before and after noisy calibration/registration of the MicroScribe G2X. The 95% confidence interval for each mean is provided.

5.7.2.3 PUMA 560

The calibrated model of the PUMA 560 that was derived from the noisy simulation is presented in Table 5.24 (using the original DH convention). In addition, the true model and the perturbed model (upon which the calibration was run) are included for comparison. The terms with the superscript "var" represent variable joint angles. The parameters that are shaded in gray were found to be "ineffective"; they could not be calibrated with the data used by the fixed-point calibration technique.

	Joint, i	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
PUMA 560 True Model	1	1.5708	0.000	$\theta_1^{\text{var}} + 0.0000$	0.000
	2	0.0000	431.800	$\theta_2^{\text{var}} + 0.0000$	0.000
	3	–1.5708	20.300	$\theta_3^{\text{var}} + 0.0000$	150.050
	4	1.5708	0.000	$\theta_4^{\text{var}} + 0.0000$	431.800
	5	–1.5708	0.000	$\theta_5^{\text{var}} + 0.0000$	0.000
	6	0.0000	0.000	$\theta_6^{\text{var}} + 0.0000$	0.000
	Joint, i	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
PUMA 560 Perturbed Model	1	1.6708	5.000	$\theta_1^{\text{var}} + 0.0000$	–10.000
	2	0.2000	446.800	$\theta_2^{\text{var}} + 0.0000$	–5.000
	3	–1.8208	30.300	$\theta_3^{\text{var}} + 0.0000$	145.050
	4	1.3708	–15.000	$\theta_4^{\text{var}} + 0.0000$	441.800
	5	–1.4708	5.000	$\theta_5^{\text{var}} + 0.0000$	10.000
	6	–0.2000	–10.000	$\theta_6^{\text{var}} + 0.0000$	–10.000
	Joint, i	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
PUMA 560 Calibrated Model (Noisy)	1	1.5710	0.067	$\theta_1^{\text{var}} + 0.0000$	–10.000
	2	0.0004	457.613	$\theta_2^{\text{var}} + 0.0001$	–9.145
	3	–1.5266	11.380	$\theta_3^{\text{var}} - 0.0222$	147.995
	4	1.0969	0.000	$\theta_4^{\text{var}} - 0.3000$	458.530
	5	–1.7708	0.008	$\theta_5^{\text{var}} + 0.3000$	0.039
	6	–0.2000	–0.076	$\theta_6^{\text{var}} + 0.2997$	–0.198

Table 5.24: Denavit–Hartenberg parameters of the PUMA 560 for its true model, the perturbed model, and the calibrated model generated from the noisy simulation. The parameters shaded in gray could not be calibrated.

The registration parameters (for translation, rotation, and scaling of the calibrated model) computed by the registration algorithm (using noisy joint values) are found in Table 5.25. The values at the bottom indicate the overall changes in robot position and orientation effected by the parameters.

Translation			**Rotation**			**Scale**
Δx (mm)	Δy (mm)	Δz (mm)	φ (rad)	θ (rad)	ψ (rad)	s
0.074	−0.282	9.327	−0.4021	0.0006	0.4023	0.9434
Change in Position			**Change in Orientation**			
9.332 mm			0.0007 rad			

Table 5.25: Registration parameters and their overall effects from the noisy post-calibration registration of the PUMA 560.

The mean position error and mean orientation error of the end-effector both before and after calibration/registration are displayed in Table 5.26. The errors were derived from the simulation of 1000 random joint configurations. Using this information, the 95% confidence interval around each mean was calculated to indicate the reliability of the estimated (mean) error.

Model	**Mean Position Error** (mm)	**Mean Orientation Error** (rad)
PUMA Perturbed	129.312 ± 4.047	0.4150 ± 0.0106
PUMA Calibrated	0.520 ± 0.013	0.6780 ± 0.0150

Table 5.26: Comparison of the mean errors of the end-effector's position and orientation before and after noisy calibration/registration of the PUMA 560. The 95% confidence interval for each mean is provided.

5.7.2.4 Stanford Arm

The calibrated model of the Stanford Arm that was derived from the noisy simulation is presented in Table 5.27 (using the original DH convention). In addition, the true model and the perturbed model (upon which the calibration was run) are included for comparison. The terms with the superscript "var" represent variable joint values. The parameters that are shaded in gray were found to be "ineffective"; they could not be calibrated with the data used by the fixed-point calibration technique.

	Joint, i	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
Stanford Arm True Model	1	–1.5708	0.000	$\theta_1^{\text{var}} + 0.0000$	412.000
	2	1.5708	0.000	$\theta_2^{\text{var}} + 0.0000$	154.000
	3	0.0000	0.000	–1.5708	$d_3^{\text{var}} + 0.000$
	4	–1.5708	0.000	$\theta_4^{\text{var}} + 0.0000$	0.000
	5	1.5708	0.000	$\theta_5^{\text{var}} + 0.0000$	0.000
	6	0.0000	0.000	$\theta_6^{\text{var}} + 0.0000$	263.000
	Joint, i	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
Stanford Arm Perturbed Model	1	–1.4708	5.000	$\theta_1^{\text{var}} + 0.0000$	402.000
	2	1.7708	15.000	$\theta_2^{\text{var}} + 0.0000$	149.000
	3	–0.2500	10.000	–1.6708	$d_3^{\text{var}} + 0.000$
	4	–1.7708	–15.000	$\theta_4^{\text{var}} + 0.0000$	10.000
	5	1.6708	5.000	$\theta_5^{\text{var}} + 0.0000$	10.000
	6	–0.2000	–10.000	$\theta_6^{\text{var}} + 0.0000$	253.000
	Joint, i	α_i (rad)	a_i (mm)	θ_i (rad)	d_i (mm)
Stanford Arm Calibrated Model (Noisy)	1	–1.5703	0.028	$\theta_1^{\text{var}} + 0.0000$	402.000
	2	1.5707	0.504	$\theta_2^{\text{var}} - 0.0001$	155.628
	3	0.0007	1.382	–1.8621	$d_3^{\text{var}} - 0.534$
	4	–1.5702	0.000	$\theta_4^{\text{var}} + 0.2918$	0.979
	5	1.5941	24.400	$\theta_5^{\text{var}} - 0.0923$	6.090
	6	–0.2000	0.000	$\theta_6^{\text{var}} + 0.0000$	262.026

Table 5.27: Denavit–Hartenberg parameters of the Stanford Arm for its true model, the perturbed model, and the calibrated model generated from the noisy simulation. The parameters shaded in gray could not be calibrated.

The registration parameters (for translation, rotation, and scaling of the calibrated model) computed by the registration algorithm (using noisy joint values) are found in Table 5.28. The values at the bottom indicate the overall changes in robot position and orientation effected by the parameters.

Translation			**Rotation**			**Scale**
Δx (mm)	Δy (mm)	Δz (mm)	φ (rad)	θ (rad)	ψ (rad)	s
−0.009	−0.536	10.203	−3.0358	0.0003	3.0361	0.9995
Change in Position			**Change in Orientation**			
10.217 mm			0.0004 rad			

Table 5.28: Registration parameters and their overall effects from the noisy post-calibration registration of the Stanford Arm.

The mean position error and mean orientation error of the end-effector both before and after calibration/registration are displayed in Table 5.29. The errors were derived from the simulation of 1000 random joint configurations. Using this information, the 95% confidence interval around each mean was calculated to indicate the reliability of the estimated (mean) error.

Model	**Mean Position Error** (mm)	**Mean Orientation Error** (rad)
Stanford Arm Perturbed	147.667 ± 4.534	0.4259 ± 0.0104
Stanford Arm Calibrated	0.759 ± 0.017	0.2134 ± 0.0041

Table 5.29: Comparison of the mean errors of the end-effector's position and orientation before and after noisy calibration/registration of the Stanford Arm. The 95% confidence interval for each mean is provided.

5.7.3 Real Robot Results

The results of the real robot experiment (described under section 5.6.5) are found below. The results are grouped according to each of the two robots used in the experiment. The average execution time of the calibration algorithms for each calibration of the robots was 20.6 minutes on the test computer (described in section 5.6.1).

5.7.3.1 AESOP 1000

The results of the four leave-one-out calibrations of the AESOP 1000 are presented in Figure 5.6. For each fixed point, the error of the initial model was calculated (shown in blue), as was the error of the model that was calibrated without access to that point's data (shown in red). The error was defined as the mean of the deviations of the calculated end-effector position from each estimated fixed-point location (per section 5.6.5.6). Approximately 20 recorded joint configurations per fixed point were used to determine the mean deviations.

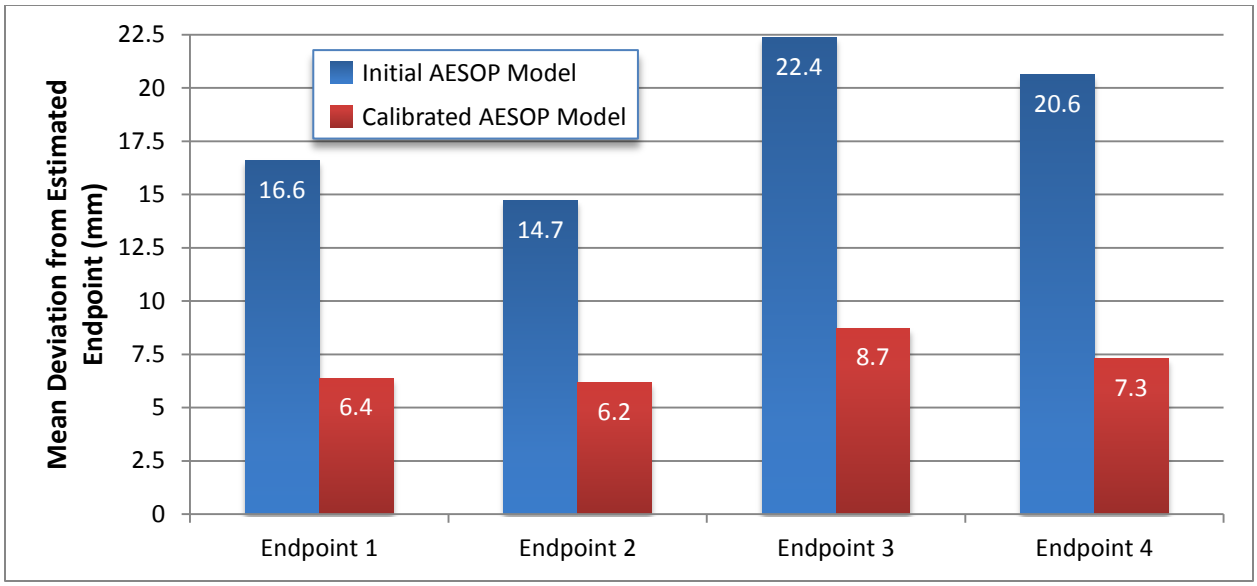

Figure 5.6: Comparison of the mean deviations of the end-effector from each estimated fixed-point location before and after four different leave-one-out calibrations of the AESOP 1000.

Table 5.30 summarizes the errors seen before and after the four leave-one-out calibrations. It provides the average values of the end-effector's mean deviations from the estimated fixed-point locations (as detailed in Figure 5.6).

Model	**Average of Mean Deviations** (mm)
AESOP Initial	18.566
AESOP Calibrated	7.152

Table 5.30: Average value of the mean deviations of the end-effector from each estimated fixed-point location before and after the four leave-one-out calibrations of the AESOP 1000.

As an example, the calculated end-effector positions of the initial AESOP model and the calibrated AESOP model for fixed point 4 are shown in Figure 5.7. The estimated fixed-point locations for both the initial and calibrated models have been placed at the origin so the spread of the end-effector positions can be compared.

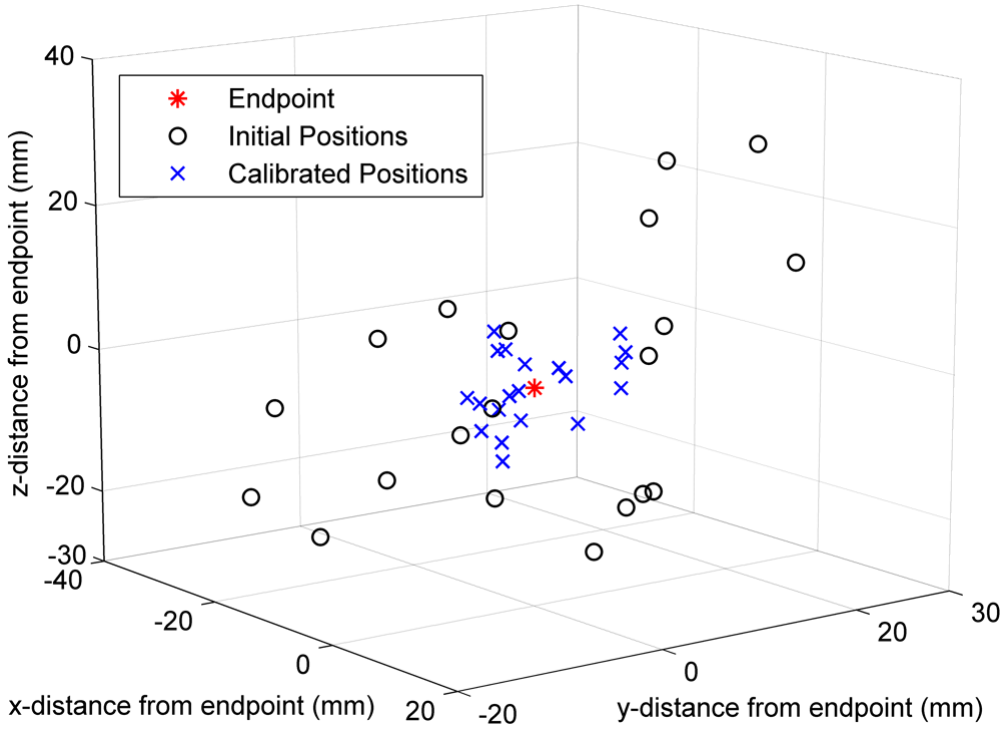

Figure 5.7: Comparison of the calculated end-effector positions before and after calibration of the AESOP 1000 model for fixed point 4. The estimated fixed-point locations have been placed at the origin.

5.7.3.2 MicroScribe G2X

The results of the five leave-one-out calibrations of the MicroScribe G2X are presented in Figure 5.8. For each fixed point, the error of the initial model was calculated (shown in blue), as was the error of the model that was calibrated without access to that point's data (shown in red). The error was defined as the mean of the deviations of the calculated end-effector position from each estimated fixed-point location (per section 5.6.5.6). Between 14 and 22 recorded joint configurations per fixed point were used to determine the mean deviations.

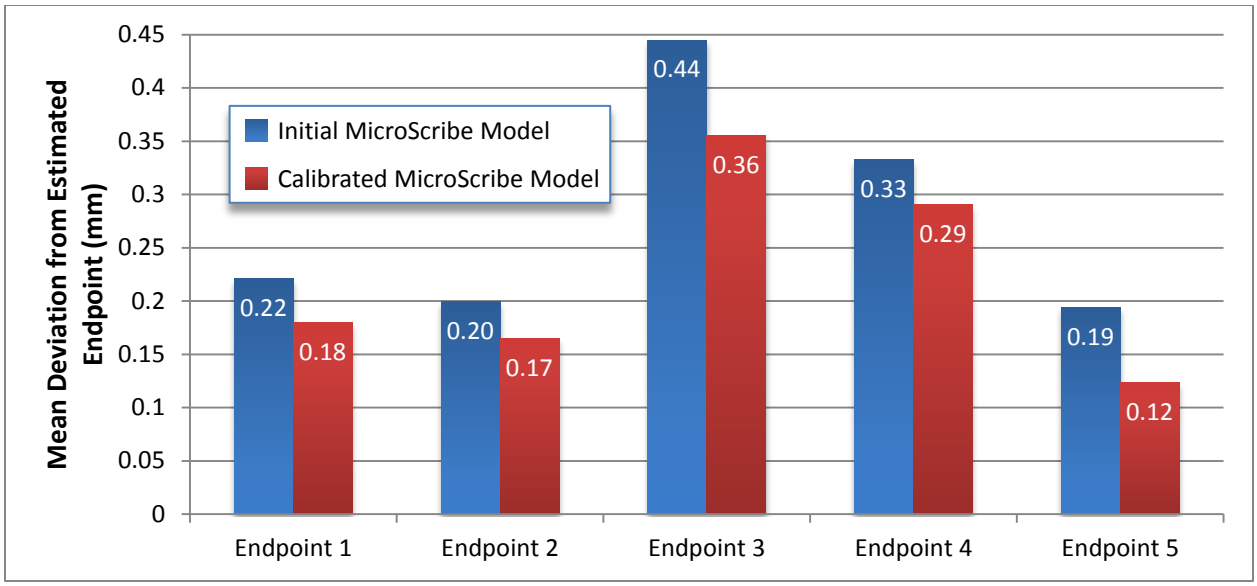

Figure 5.8: Comparison of the mean deviations of the end-effector from each estimated fixed-point location before and after five different leave-one-out calibrations of the MicroScribe G2X.

Table 5.31 summarizes the errors seen before and after the five leave-one-out calibrations. It provides the average values of the end-effector's mean deviations from the estimated fixed-point locations (as detailed in Figure 5.8).

Model	**Average of Mean Deviations** (mm)
MicroScribe Initial	0.278
MicroScribe Calibrated	0.223

Table 5.31: Average value of the mean deviations of the end-effector from each estimated fixed-point location before and after the five leave-one-out calibrations of the MicroScribe G2X.

5.8 Discussion

5.8.1 Ideal Simulation Discussion

The results of the ideal simulation experiment were excellent for all four robots. In all cases, the calibration/registration algorithms were able to reduce the mean position error to practically zero. This was possible due to the lack of noise in the simulated data. The only sources of (very insignificant) position error seemed to be the finite numerical precision and finite execution time available to the algorithms.

The mean orientation error was also significantly reduced (but still meaningfully present) for each of the robots. This was interesting because the presented calibration/registration algorithms do not have the ability to directly optimize the orientation error (due to the lack of orientation data). It turned out that by optimizing for the position error, the algorithms generated models for these robots that were also closer to producing the ideal orientations. Note that the orientation error could have increased under different circumstances, so the presented algorithms should not be relied upon to reduce orientation error.

It is also interesting to compare the parameters of each calibrated DH model to the parameters of the corresponding true model. Despite the practically nonexistent position error, the majority of the calibrated parameters do not match the true parameters. Only 6 out of 28 parameters (AESOP 1000), 7 out of 24 parameters (MicroScribe G2X), 13 out of 24 parameters (PUMA 560), and 10 out of 24 parameters (Stanford Arm) actually matched (within the precision of the presented DH tables). Moreover, 4 of the PUMA 560 parameters and 2 of the Stanford Arm parameters only matched because the fixed joint offsets for those robots were never perturbed.

There are two reasons why most of the calibrated parameters did not match the true parameters even though the position error was practically zero. First, certain DH parameters cannot be calibrated due to the lack of absolute end-effector positional coordinates and the absence of orientation data. As described under section 5.4.4, these "ineffective" parameters have no effect on the calibration technique's error metric. Fortunately, the registration process compensates for this problem (with respect to the position error). The translation, rotation, and scaling provided by the registration parameters enabled the correct absolute positions to be achieved despite the fact that some DH parameters were left uncalibrated.

The second reason why most of the parameters did not match is because a DH model is often not an ideal, minimal representation of a robot. As explained under section 5.5, there can be poorly defined or redundant DH parameters, which means there are multiple sets of parameters that effectively produce the same robot. For example, the presence of prismatic joints or consecutive parallel joint axes will lead to such problems. When coupled with the lack of orientation data in the calibration process, the result is that it is often not possible to exactly identify all the parameters of a specific true model. All of the chosen true models have one or more of these issues.

The registration parameters that were computed for each calibrated robot model successfully compensated for the lack of absolute coordinates for the end-effector's position. The proof is that the position errors were practically zero even though a number of DH parameters could not be calibrated. Without the registration transformations, the mean position errors would be significant.

The translational registration parameters show that three robots each needed about 1 cm of repositioning, although the AESOP required over 5 cm (in part due to having more uncalibrated length parameters). The rotation parameters indicate that the AESOP and MicroScribe needed approximately 0.40 rad and 0.22 rad of reorienting (respectively), whereas the PUMA and Stanford Arm required none (because their fixed joint offsets were never perturbed). The scale parameters show that the robots with prismatic joints (the AESOP and Stanford Arm) needed no scale adjustment, but the other robots required a slight scale correction (approximately 1%).

5.8.2 Noisy Simulation Discussion

The results of the noisy simulation experiment were very good for all four robots. The calibration/registration algorithms were able to reduce each mean position error to a value

between 0.308 and 1.202 mm. Obviously, the addition of noise to the joint feedback led to this error, which was not seen in the ideal simulation experiment. The noise added to each joint value was between ±1 mrad or ±1 mm, so the magnitude of the mean position errors seems reasonable.

The mean orientation error was also significantly reduced (but still meaningfully present) for all the robots except the PUMA 560. For the PUMA, the mean orientation error increased from 0.4150 rad to 0.6780 rad. There are several factors that explain the increase. First, the presented calibration/registration algorithms do not have the ability to directly optimize the orientation error (due to the lack of orientation data). In addition, half of the PUMA model's six joints only affect the orientation of the end-effector, and the noise in the joint feedback made the corresponding DH parameters difficult to detect as "ineffective" (per section 5.4.4). Thus, more orientation error was expected for the PUMA. For the other three robots, the decrease in orientation error was comparable to the results of the ideal simulation experiment.

Very few parameters of the calibrated DH models matched the parameters of the corresponding true models (within the precision of the presented DH tables). In fact, only 2 out of 24 parameters (MicroScribe G2X), 2 out of 24 parameters (PUMA 560), and 4 out of 24 parameters (Stanford Arm) actually matched, and none of the 28 AESOP 1000 parameters matched. Moreover, 1 of the PUMA 560 parameters and 2 of the Stanford Arm parameters only matched because the fixed joint offsets for those robots were never perturbed. Nevertheless, for the calibrated parameters that matched in the ideal experiment, the same parameters in the noisy experiment were relatively close to their true values.

The most obvious cause of the parameter mismatch was the noise added to the joint feedback, which made it extremely difficult to exactly recover any of the true parameters. In

addition, the other causes of parameter mismatch described in the previous section (uncalibrated parameters and non-ideal DH models) also apply to this experiment. In fact, the noise made it even harder to detect parameters that could not be calibrated (i.e., were ineffective), as explained below.

The calibration algorithms attempt to detect parameters that cannot be calibrated by perturbing each parameter and checking for no change in the output position error (per section 5.4.4.2). Unfortunately, this process is dependent upon the accuracy of the model. Errors in the partially calibrated models (caused by the added noise) made the algorithms miss a few parameters that were detected as "ineffective" in the ideal simulation (for the AESOP 1000 and PUMA 560). Consequently, the algorithms in the noisy simulation tried to calibrate a few extra parameters, which mostly ended up further from the true values than their initial values.

The registration parameters that were computed for each calibrated robot model appear to have successfully compensated for the lack of absolute coordinates for the end-effector's position. The final mean position errors were much smaller than the amounts by which each model was translated/rotated/scaled by the registration parameters, indicating that the registration transformed the models in at least a mostly correct manner. In addition, the registration parameters calculated in the noisy simulation were mostly similar to those of the ideal simulation, and the differences appear to reflect the changes between the experiments. Note that the registration process can't correct internal errors of the DH model (caused by the added joint noise), which explains why some position error remained after registration.

The translational registration parameters show that three robots each needed roughly 1 cm of repositioning, although the AESOP required almost 6 cm (in part due to having more uncalibrated length parameters). The rotation parameters indicate that the AESOP and

MicroScribe needed approximately 0.70 rad and 0.22 rad of reorienting (respectively), whereas the PUMA and Stanford Arm required almost none (because their fixed joint offsets were never perturbed). The scale parameters show that the robots with prismatic joints (the AESOP and Stanford Arm) needed almost no scale adjustment, but the MicroScribe and PUMA required some small scale corrections (about 2% and 5.7%, respectively).

5.8.3 Real Robot Discussion

The results of the real robot experiment for the AESOP 1000 were very good. As seen in Table 5.30, the estimated mean position error of the robot was reduced by more than half (from 1.86 cm to 0.72 cm). In addition, Figure 5.6 shows that the reduction was relatively consistent for each of the four fixed points and calibrated models. This is important because each calibrated model was generated without access to the fixed-point data (joint configurations) against which it was tested. These facts indicate that the calibration technique is correctly approximating the actual structure of the robot instead of generating a model that only fits the test data.

Figure 5.7 gives a graphical representation of the increased precision provided by the calibration technique. After the calibration, there is an obvious decrease in the spread of the calculated end-effector positions from the estimated fixed-point locations. Assuming the robot can be accurately registered with little scale adjustment, the figure demonstrates the improved position accuracy that is possible with the presented calibration technique.

The large improvement for the AESOP 1000 was anticipated due to the expectation of imperfections in the initial model. As explained in section 5.6.5.3, the starting model for the AESOP was generated from older, manual measurements of the robot. This left a lot of room for improvement, even with the noise and other sources of error in the real joint measurements.

Further gains in the accuracy might not have been possible because the AESOP 1000 was originally designed for manual positioning, not precise computer control.

For the MicroScribe G2X, the real robot experiment also produced good results. As shown in Table 5.31, the estimated mean position error of the robot was reduced by approximately 20% (from 0.28 mm to 0.22 mm). According to Figure 5.8, a decrease in error was seen for all five fixed points and calibrated models. Since each calibrated model was generated without access to the fixed-point data against which it was tested, the calibration technique seems to be producing models that correctly approximate the actual structure of the robot.

The improvement for the MicroScribe G2X was much less than the improvement seen for the AESOP 1000. This was expected because the initial model for the MicroScribe was already relatively accurate. Thus, there was less room for improvement, and the noise/error in the real joint measurements was likely limiting any further progress.

5.9 Conclusion

The results show that the presented calibration technique was able to increase the positional accuracy of the end-effector for robot models in each of the three experiments. Positive outcomes were achieved for all four robots that were tested, which demonstrates the flexibility of the technique. Not all of the true DH parameters (or approximations thereof) could be recovered via the calibration, but this was expected due to limitations in the collected data and dependent/redundant/ambiguous parameters inherent in DH modeling. Ultimately, the calibration technique generated models that effectively reduced the positional error of the end-effector.

5.10 Future Work

There are many ways in which the presented calibration technique could be improved. For example, more intelligent scaling of the pose components and DH parameters could result in improved performance [109, 111]. An automated method of selecting fixed-point locations within the robot's workspace could also improve the calibration process [109, 111].

The calibration technique currently uses a direct search method that requires no Jacobian matrices to be calculated. This enables the technique to be implemented more easily and adapted to alternative robot modeling conventions with less effort, but it also means that advantages of a Jacobian-based technique are lost. For example, a Jacobian-based technique will generally be faster than a direct search technique.

An analysis of the identification Jacobian would provide more information about the identifiability of various DH parameters [105, 109-112]. The unidentifiable and weakly identifiable parameters could be found more accurately. In addition, linearly dependent sets of parameters could be located. The technique would be benefit if it were able to remove most of the linearly dependent parameters.

To better model the uncertainties of parameters and measurement errors, *a priori* estimates for these quantities could be incorporated into the technique. For example, the implicit loop method provides a powerful way to incorporate and test statistical variables in the calibration [122]. The incorporation of statistical methods can improve the robustness of the technique and help identify unmodeled factors.

There are a number of robot parameters beyond the four DH parameters that define aspects of the robot. For example, it may be necessary to calibrate the gains of the joint feedback sensors. In some cases, it is desired to work with parameters of the Hayati convention to avoid

ambiguities in the robot model [111, 134]. In addition, there are a large number of dynamic/inertial parameters that need to be calibrated for certain applications. Some of these kinds of parameters could be added to the technique if found to be compatible.

Additional testing of the real robots (AESOP 1000 and MicroScribe G2X) is expected to be performed. The robots can be registered to different objects, and the accuracy can be assessed with an external measurement device (e.g., an optical tracker). The calibrated robot models should also be tested with the image guidance system of Chapter 4.

Chapter 6:
Summary and Future Work

6.1 Summary

In Chapter 3, an open software system for processing, analyzing, and classifying Raman spectra was introduced. The system was successfully applied to the classification of two Raman data sets. In Chapter 4, a novel image guidance system for a robot with an integrated sensor was presented. The system was able to successfully track the sensor, collect new data, classify the data, and present the results in near real time. Finally, in Chapter 5, a flexible kinematic calibration technique for serial manipulators was described. The technique was able to improve the positional accuracy of four different robots in three different experiments.

6.2 Future Work

As discussed in Chapter 3, Chapter 4, and Chapter 5, there are number of potential improvements that can be made to the systems and techniques that were developed in this research. However, from a broader perspective, the goal of this work was to make contributions that facilitate the use of sensor-integrated robotic systems. In fact, the chosen contributions represent important building blocks that could be used to create an actual image-guided robotic system with an integrated Raman probe. For example, they could be merged to produce a powerful robotic surgery system, as explained below.

The developed Raman software system (Chapter 3) provides the necessary infrastructure to identify diseased (e.g., cancerous) human tissue. The work described in [104] will make it possible to move a Raman probe with a surgical robot (the AESOP 1000) inside the body. The image guidance system (Chapter 4) offers a way to track the Raman probe inside the body,

initiate scans, and view the results in real time. Lastly, the kinematic calibration technique (Chapter 5) can be used to improve the limited accuracy of the AESOP, which was noted in [104].

The aforementioned systems were designed with the goal of eventually working together. However, some of them have not been tested together, and there is still some refinement and application-specific work that has to be done. Some of the important high-level tasks that are needed to produce a functional combined system are described below.

First, the full kinematic calibration technique (including the registration process) should be applied to the AESOP 1000 using new joint configuration data with an attached Raman probe. Second, the Raman classification software must be trained to recognize various tissues or other specimens of interest. It may also be necessary to adapt more of the Raman software algorithms for use with the real-time image guidance system. Third, the control system for the robot should be tested and possibly adjusted to ensure that the Raman probe can be maneuvered with sufficient ease and precision. Fourth, the Raman probe should be tested and potentially altered to work in its intended environment (e.g., in the body). Fifth, imaging data of the target environment has to be obtained and processed.

Many other steps would likely need to be completed before the combined system could be used in a particular application. For example, if the system were intended to be used inside a person during surgery, extensive testing and refinement would be required to ensure the system is safe and effective. In some cases, other instruments may need to be tracked, and domain-specific information may need to be displayed within the image guidance system. Meeting these kinds of requirements may take considerable effort, but the work is much easier when systems like those developed in this research are available.

A future image-guided robotic surgery system with an integrated Raman probe could be a great benefit in the operating room. The surgeon could diagnose diseased tissue in seconds instead of waiting for a pathological analysis. Consequently, the surgeon could immediately remove the diseased tissue while preserving the surrounding healthy tissue. This kind of technology has the potential to save lives and reduce the costs of operations.

REFERENCES

[1] J. P. Grotzinger, J. Crisp, A. R. Vasavada, R. C. Anderson, C. J. Baker, R. Barry*, et al.*, "Mars Science Laboratory mission and science investigation," *Space Science Reviews,* pp. 1-52, 2012.

[2] L. Robledo, M. Carrasco, and D. Mery, "A survey of land mine detection technology," *International Journal of Remote Sensing,* vol. 30, pp. 2399-2410, 2009.

[3] A. M. Okamura, "Haptic feedback in robot-assisted minimally invasive surgery," *Current opinion in urology,* vol. 19, p. 102, 2009.

[4] A. K. Pandya, G. K. Serhatkulu, A. Cao, R. E. Kast, H. Dai, R. Rabah*, et al.*, "Evaluation of Pancreatic Cancer With Raman Spectroscopy in a Mouse Model," *Pancreas,* vol. 36, pp. e1-e8, 2008.

[5] H. Wills, R. Kast, C. Stewart, R. Rabah, A. Pandya, J. Poulik*, et al.*, "Raman spectroscopy detects and distinguishes neuroblastoma and related tissues in fresh and (banked) frozen specimens," *Journal of Pediatric Surgery,* vol. 44, pp. 386-391, 2009.

[6] H. Wills, R. Kast, C. Stewart, B. Sullivan, R. Rabah, J. Poulik*, et al.*, "Diagnosis of Wilms' tumor using near-infrared Raman spectroscopy," *Journal of Pediatric Surgery,* vol. 44, pp. 1152-1158, 2009.

[7] J. C. Carter, S. M. Angel, M. Lawrence-Snyder, J. Scaffidi, R. E. Whipple, and J. G. Reynolds, "Standoff Detection of High Explosive Materials at 50 Meters in Ambient Light Conditions Using a Small Raman Instrument," *Applied Spectroscopy,* vol. 59, pp. 769-775, 2005.

[8] G. B. Courrèges-Lacoste, B. Ahlers, and F. R. Pérez, "Combined Raman spectrometer/laser-induced breakdown spectrometer for the next ESA mission to Mars,"

Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy, vol. 68, pp. 1023-1028, 2007.

[9] S. Y.-S. Lee, "An augmented reality interface for multi-robot tele-operation and control," Ph.D. Thesis, Electrical Engineering, Wayne State University, ETD Collection, 2011.

[10] K. Chintamani, "Augmented reality navigation interfaces improve human performance in end-effector controlled telerobotics," Ph.D. Thesis, Mechanical Engineering, Wayne State University, ETD Collection, 2010.

[11] M. F. Mrozek, D. Zhang, and D. Ben-Amotz, "Oligosaccharide identification and mixture quantification using Raman spectroscopy and chemometric analysis," *Carbohydrate Research,* vol. 339, pp. 141-145, 2004.

[12] C. J. Strachan, D. Pratiwi, K. C. Gordon, and T. Rades, "Quantitative analysis of polymorphic mixtures of carbamazepine by Raman spectroscopy and principal components analysis," *Journal of Raman Spectroscopy,* vol. 35, pp. 347-352, 2004.

[13] A. S. Haka, Z. Volynskaya, J. A. Gardecki, J. Nazemi, J. Lyons, D. Hicks*, et al.*, "In vivo Margin Assessment during Partial Mastectomy Breast Surgery Using Raman Spectroscopy," *Cancer Research,* vol. 66, pp. 3317-3322, March 15 2006.

[14] J. T. Motz, S. J. Gandhi, O. R. Scepanovic, A. S. Haka, J. R. Kramer, R. R. Dasari*, et al.*, "Real-time Raman system for in vivo disease diagnosis," *Journal of Biomedical Optics,* vol. 10, p. 031113, 2005.

[15] T. C. Bakker Schut, R. Wolthuis, P. J. Caspers, and G. J. Puppels, "Real-time tissue characterization on the basis of in vivo Raman spectra," *J. Raman Spectroscopy,* vol. 33, pp. 580-585, 2002.

[16] P. J. Caspers, G. W. Lucassen, and G. J. Puppels, "Combined In Vivo Confocal Raman Spectroscopy and Confocal Microscopy of Human Skin," *Biophysical Journal,* vol. 85, pp. 572-580, 2003.

[17] A. Mahadevan-Jansen, M. F. Mitchell, N. Ramanujam, U. Utzinger, and R. Richards-Kortum, "Development of a fiber optic probe to measure NIR Raman spectra of cervical tissue in vivo," *Photochemistry and Photobiology,* vol. 68, pp. 427-431, September 1998.

[18] A. Robichaux-Viehoever, E. Kanter, H. Shappell, D. Billheimer, H. Jones Iii, and A. Mahadevan-Jansen, "Characterization of Raman Spectra Measured in Vivo for the Detection of Cervical Dysplasia," *Applied Spectroscopy,* vol. 61, pp. 986-993, 2007.

[19] B. W. King, L. A. Reisner, R. D. Ellis, M. D. Klein, G. W. Auner, and A. K. Pandya, "Optimized port placement for *in vivo* biosensors," *The International Journal of Medical Robotics and Computer Assisted Surgery,* vol. 5, pp. 267-275, 2009.

[20] L. A. Reisner, B. W. King, M. D. Klein, G. W. Auner, and A. K. Pandya, "A prototype biosensor-integrated image-guided surgery system," *The International Journal of Medical Robotics and Computer Assisted Surgery,* vol. 3, pp. 82-88, 2007.

[21] I. Salakhutdinov and G. W. Auner, "Diffraction gratings for compact integrated optical structures," in *Proceedings of SPIE*, 2009, p. 72050O.

[22] S. Sigurdsson, P. A. Philipsen, L. K. Hansen, J. Larsen, M. Gniadecka, and H. C. Wulf, "Detection of skin cancer by classification of Raman spectra," *IEEE Transactions on Biomedical Engineering,* vol. 51, pp. 1784-1793, 2004.

[23] A. S. Haka, K. E. Shafer-Peltier, M. Fitzmaurice, J. Crowe, R. R. Dasari, and M. S. Feld, "Diagnosing breast cancer by using Raman spectroscopy," *Proc National Academy of Sciences of the USA,* vol. 102, pp. 12371-12376, 2005.

[24] U. Utzinger, D. L. Heintzelman, A. Mahadevan-Jansen, A. Malpica, M. Follen, and R. Richards-Kortum, "Near-infrared Raman spectroscopy for in vivo detection of cervical precancers," *Appl. Spectroscopy,* vol. 55, pp. 955-959, 2001.

[25] N. Stone, C. Kendall, N. Shepherd, P. Crow, and H. Barr, "Near-infrared Raman spectroscopy for the classification of epithelial pre-cancers and cancers," *Journal of Raman Spectroscopy,* vol. 33, pp. 564-573, 2002.

[26] A. S. Haka, K. E. Shafer-Peltier, M. Fitzmaurice, J. Crowe, R. R. Dasari, and M. S. Feld, "Identifying Microcalcifications in Benign and Malignant Breast Lesions by Probing Differences in Their Chemical Composition Using Raman Spectroscopy," vol. 62, pp. 5375-5380, 2002.

[27] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *Journal of Applied Mechanics,* vol. 22, pp. 215-221, 1955.

[28] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed. Prentice Hall, 2004.

[29] A. Jemal, F. Bray, M. M. Center, J. Ferlay, E. Ward, and D. Forman, "Global cancer statistics," *CA: a cancer journal for clinicians,* vol. 61, pp. 69-90, 2011.

[30] J. R. Ferraro and K. Nakamoto, *Introductory Raman Spectroscopy*, Academic Press, 2003.

[31] M. J. Pelletier, *Analytical Applications of Raman Spectroscopy*, Blackwell Publishing, 1999.

[32] C. Fantini, A. Jorio, M. Souza, M. S. Strano, M. S. Dresselhaus, and M. A. Pimenta, "Optical Transition Energies for Carbon Nanotubes from Resonant Raman Spectroscopy: Environment and Temperature Effects," *Physical Review Letters,* vol. 93, p. 147406, 2004.

[33] S. Piscanec, M. Cantoro, A. C. Ferrari, J. A. Zapien, Y. Lifshitz, S. T. Lee*, et al.*, "Raman spectroscopy of silicon nanowires," *Physical Review B,* vol. 68, p. 241312, 2003.

[34] P. C. A. M. Buijtels, H. F. M. Willemse-Erix, P. L. C. Petit, H. P. Endtz, G. J. Puppels, H. A. Verbrugh*, et al.*, "Rapid identification of mycobacteria by Raman spectroscopy," *J. Clinical Microbiology,* pp. 01763-07, January 3, 2008 2008.

[35] K. T. Tsen, E. C. Dykeman, O. F. Sankey, S. W. D. Tsen, N. T. Lin, and J. G. Kiang, "Raman scattering studies of the low-frequency vibrational modes of bacteriophage M13 in water—observation of an axial torsion mode," *Nanotechnology,* vol. 17, pp. 5474-5479, 2006.

[36] S. J. Tinnemans, M. H. F. Kox, T. A. Nijhuis, T. Visser, and B. M. Weckhuysen, "Real time quantitative Raman spectroscopy of supported metal oxide catalysts without the need of an internal standard," *Physical Chemistry Chemical Physics,* vol. 7, pp. 211-216, 2005.

[37] B. M. Vogelaar, A. D. van Langeveld, S. Eijsbouts, and J. A. Moulijn, "Analysis of coke deposition profiles in commercial spent hydroprocessing catalysts using Raman spectroscopy," *Fuel,* vol. 86, pp. 1122-1129, 2007.

[38] L. P. Choo-Smith, H. G. M. Edwards, H. P. Endtz, J. M. Kros, F. Heule, H. Barr*, et al.*, "Medical applications of Raman spectroscopy: from proof of principle to clinical implementation," *Biopolymers (Biospectroscopy),* vol. 67, pp. 1–9, 2002.

[39] H. P. Buschman, J. T. Motz, G. Deinum, T. J. Römer, M. Fitzmaurice, J. R. Kramer*, et al.*, "Diagnosis of human coronary atherosclerosis by morphology-based Raman spectroscopy," *Cardiovascular Pathology,* vol. 10, pp. 59-68, 2001.

[40] A. Mahadevan-Jansen and R. Richards-Kortum, "Raman spectroscopy for cancer detection: a review," in *19th International Conference of IEEE/EMBS*, Chicago, IL, 1997, pp. 2722-2728.

[41] E. B. Hanlon, R. Manoharan, T. W. Koo, K. E. Shafer, J. T. Motz, M. Fitzmaurice*, et al.*, "Prospects for in vivo Raman spectroscopy," *Physics in Medicine and Biology,* vol. 45, p. 1–59, 2000.

[42] M. Hashimoto, T. Araki, and S. Kawata, "Molecular vibration imaging in the fingerprint region by use of coherent anti-Stokes Raman scattering microscopy with a collinear configuration," *Optics Letters,* vol. 25, pp. 1768-1770, 2000.

[43] D. L. Massart, *Handbook of Chemometrics and Qualimetrics: Part A*, Elsevier, 1997.

[44] T. Howley, M. G. Madden, M. L. O'Connell, and A. G. Ryder, "The Effect of Principal Component Analysis on Machine Learning Accuracy with High Dimensional Spectral Data," *Knowledge-Based Systems,* vol. 19, pp. 363-370, 2006.

[45] A. G. Ryder, G. M. O'Connor, and T. J. Glynn, "Quantitative analysis of cocaine in solid mixtures using Raman spectroscopy and chemometric methods," *Journal of Raman Spectroscopy,* vol. 31, pp. 221-227, 2000.

[46] R. N. Feudale, N. A. Woody, H. Tan, A. J. Myles, S. D. Brown, and J. Ferré, "Transfer of multivariate calibration models: a review," *Chemometrics and Intelligent Laboratory Systems,* vol. 64, pp. 181-192, 2002.

[47] O. E. De Noord, "Multivariate Calibration Standardization," *Chemometrics and Intelligent Laboratory Systems,* vol. 25, pp. 85-97, 1994.

[48] K. E. Shafer-Peltier, A. S. Haka, M. Fitzmaurice, J. Crowe, J. Myles, R. R. Dasari*, et al.*, "Raman microspectroscopic model of human breast tissue: implications for breast cancer diagnosis in vivo," *J. Raman Spectroscopy,* vol. 33, pp. 552-563, 2002.

[49] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed. New York: Springer-Verlag, 2002.

[50] M. J. Adams, *Chemometrics In Analytical Spectroscopy*, Royal Society of Chemistry, 2004.

[51] J. E. Jackson, *A User's Guide to Principal Components*, Wiley, 1991.

[52] Y. Wang, D. J. Veltkamp, and B. R. Kowalski, "Multivariate instrument standardization," *Analytical Chemistry,* vol. 63, pp. 2750-2756, 1991.

[53] Z. Wang, T. Dean, and B. R. Kowalski, "Additive Background Correction in Multivariate Instrument Standardization," *Analytical Chemistry,* vol. 67, pp. 2379-2385, 1995.

[54] D. Michie, D. J. Spiegelhalter, and C. C. Taylor, *Machine Learning, Neural and Statistical Classification*. Upper Saddle River, NJ: Ellis Horwood, 1994.

[55] A. Molckovsky, L.-M. Wong Kee Song, M. G. Shim, N. E. Marcon, and B. C. Wilson, "Diagnostic potential of near-infrared Raman spectroscopy in the colon: differentiating adenomatous from hyperplastic polyps," *Gastrointestinal Endoscopy,* vol. 57, pp. 396-402, March 2003.

[56] J. S. Thakur, H. Dai, G. K. Serhatkulu, R. Naik, V. M. Naik, A. Cao*, et al.*, "Raman spectral signatures of mouse mammary tissue and associated lymph nodes: normal, tumor and mastitis," *J. Raman Spectroscopy,* vol. 38, pp. 127-134, February 2007.

[57] S. J. Baek, A. Park, J. Y. Kim, S. Y. Na, Y. Won, and J. Choo, "Detection of Basal Cell Carcinoma by Automatic Classification of Confocal Raman Spectra," in *Computational Intelligence and Bioinformatics*. vol. 4115, S. Istrail, P. Pevzner, and M. Waterman, Eds., ed: Springer Berlin / Heidelberg, 2006, pp. 402-411.

[58] S. K. Majumder, N. Ghosh, and P. K. Gupta, "Support vector machine for optical diagnosis of cancer," *Journal of Biomedical Optics,* vol. 10, p. 024034, 2005.

[59] P. Legendre and L. Legendre, *Numerical Ecology*, 2nd ed. Amsterdam: Elsevier Science, 1998.

[60] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*, 1st ed. Cambridge: The MIT Press, 1995.

[61] C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery,* vol. 2, pp. 121-167, 1998.

[62] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. London: Springer, 2009.

[63] R. P. Paul, *Robot manipulators: mathematics, programming, and control*. USA: MIT Press, 1981.

[64] L. A. Reisner, A. Cao, and A. K. Pandya, "An integrated software system for processing, analyzing, and classifying Raman spectra," *Chemometrics and Intelligent Laboratory Systems,* vol. 105, pp. 83-90, January 15 2011.

[65] M. C. Demirel, P. Kao, N. Malvadkar, H. Wang, X. Gong, M. Poss*, et al.*, "Bio-organism sensing via surface enhanced Raman spectroscopy on controlled metal/polymer nanostructured substrates," *Biointerphases,* vol. 4, pp. 35-41, 2009.

[66] K. L. Brown, O. Y. Palyvoda, J. S. Thakur, S. L. Nehlsen-Cannarella, O. R. Fagoaga, S. A. Gruber*, et al.*, "Raman spectroscopic differentiation of activated versus non-activated T lymphocytes: An in vitro study of an acute allograft rejection model," *Journal of Immunological Methods,* vol. 340, pp. 48-54, 2009.

[67] K. De Gussem, J. De Gelder, P. Vandenabeele, and L. Moens, "The Biodata toolbox for MATLAB," *Chemometrics and Intelligent Laboratory Systems,* vol. 95, pp. 49-52, 2009.

[68] R. A. Viscarra Rossel, "ParLeS: Software for chemometric analysis of spectroscopic data," *Chemometrics and Intelligent Laboratory Systems,* vol. 90, pp. 72-83, 2008.

[69] A. Cao, A. K. Pandya, G. K. Serhatkulu, R. E. Weber, H. Dai, J. S. Thakur*, et al.*, "A robust method for automated background subtraction of tissue fluorescence," *Journal of Raman Spectroscopy,* vol. 38, pp. 1199-1205, September 2007.

[70] C. R. Mittermayr, S. G. Nikolov, H. Hutter, and M. Grasserbauer, "Wavelet denoising of Gaussian peaks: A comparative study," *Chemometrics and Intelligent Laboratory Systems,* vol. 34, pp. 187-202, 1996.

[71] M. J. Tosina Muñoz, "Diseño de un filtro morfológico para la reducción de fluorescencia en espectros Raman," Master's Thesis, Teoría de la Señal y Comunicaciones, Universidad Politécnica de Catalunya, 2008.

[72] R. Perez-Pueyo, M. J. Soneira, and S. Ruiz-Moreno, "Morphology-based automated baseline removal for Raman spectra of artistic pigments," *Applied spectroscopy,* vol. 64, pp. 595-600, 2010.

[73] F. Y. Shih, *Image processing and mathematical morphology: Fundamentals and applications*, CRC, 2009.

[74] N. Brandt, O. Brovko, A. Y. Chikishev, and O. Paraschuk, "Optimization of the rolling-circle filter for Raman background subtraction," *Applied spectroscopy,* vol. 60, pp. 288-293, 2006.

[75] R. J. Meier, "On art and science in curve-fitting vibrational spectra," *Vibrational spectroscopy,* vol. 39, pp. 266-269, 2005.

[76] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions," *SIAM Journal on Optimization,* vol. 9, pp. 112-147, 1998.

[77] Z. Movasaghi, S. Rehman, and D. I. U. Rehman, "Raman spectroscopy of biological tissues," *Applied Spectroscopy Reviews,* vol. 42, pp. 493-541, 2007.

[78] A. Pandya, "Medical augmented reality system for image-guided and robotic surgery: Development and surgeon factors analysis," Ph.D. Thesis, Biomedical Engineering, Wayne State University, ETD Collection, 2004.

[79] F. Sauer, "Image Registration: Enabling Technology for Image Guided Surgery and Therapy," in *27th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Shanghai, China, 2005, pp. 7242-7245.

[80] P. Crow, N. Stone, C. A. Kendall, J. S. Uff, J. A. Farmer, H. Barr*, et al.*, "The use of Raman spectroscopy to identify and grade prostatic adenocarcinoma in vitro," *Br J Cancer,* vol. 89, pp. 106-8, Jul 7 2003.

[81] C. Krafft, S. Miljanic, S. B. Sobottka, G. Schackert, and R. Salzer, "Near-infrared Raman spectroscopy to study the composition of human brain tissue and tumors," *Proceedings of SPIE-The International Society for Optical Engineering,* vol. 5141, pp. 230-236, 2003.

[82] C. J. Frank, D. C. Redd, T. S. Gansler, and R. L. McCreery, "Characterization of human breast biopsy specimens with near-IR Raman spectroscopy," *Anal Chem,* vol. 66, pp. 319-26, Feb 1 1994.

[83] M. Gniadecka, H. C. Wulf, O. F. Nielsen, D. H. Christensen, and J. Hercogova, "Distinctive molecular abnormalities in benign and malignant skin lesions: studies by Raman spectroscopy," *Photochem Photobiol,* vol. 66, pp. 418-23, Oct 1997.

[84] C. K. Johansson, D. H. Christensen, and O. F. Nielsen, "Near-infrared Fourier transform Raman spectral studies of human skin," *Dansk Kemi,* vol. 80, pp. 12-13, 1999.

[85] Y.-K. Min, T. Yamamoto, E. Kohda, T. Ito, and H.-o. Hamaguchi, "1064 nm near-infrared multichannel Raman spectroscopy of fresh human lung tissues," *Journal of Raman Spectroscopy,* pp. 73–76, 2005.

[86] K. Maquelin, L. P. Choo-Smith, T. van Vreeswijk, H. P. Endtz, B. Smith, R. Bennett*, et al.*, "Raman spectroscopic method for identification of clinically relevant microorganisms growing on solid culture medium," *Anal Chem,* vol. 72, pp. 12-9, Jan 1 2000.

[87] K. S. Kalasinsky and V. F. Kalasinsky, "Infrared and Raman microspectroscopy of foreign materials in tissue specimens," *Spectrochimica Acta A: Mol Biomol Spectroscopy,* vol. 61, pp. 1707-13, May 2005.

[88] M. D. Schaeberle, V. F. Kalasinsky, J. L. Luke, E. N. Lewis, I. W. Levin, and P. J. Treado, "Raman chemical imaging: histopathology of inclusions in human breast tissue," *Anal Chem,* vol. 68, pp. 1829-33, Jun 1 1996.

[89] D. Pappas, B. W. Smith, and J. D. Winefordner, "Raman spectroscopy in bioanalysis," *Talanta,* vol. 51, pp. 131-144, 2000.

[90] A. Pandya and G. Auner, "Robotic Technology: a Journey into the Future," in *Robotic Urologic Surgery*. vol. 31, M. Menon and S. Das, Eds., ed: Elsevier, Inc., 2004, pp. 793-800.

[91] A. Lorincz, D. Haddad, R. Naik, V. Naik, A. Fung, A. Cao*, et al.*, "Raman spectroscopy for neoplastic tissue differentiation: a pilot study," *Journal of Pediatric Surgery,* vol. 39, pp. 953-956, 2004.

[92] R. D. Bucholz, K. R. Smith, K. A. Laycock, and L. L. McDurmont, "Three-dimensional localization: From image-guided surgery to information-guided therapy," *Methods,* vol. 25, pp. 186-200, Oct 2001.

[93] A. Pandya, M. R. Siadat, and G. Auner, "Design, implementation and accuracy of a prototype for medical augmented reality," *Computer Aided Surgery,* vol. 10, pp. 23-35, 2005.

[94] L. T. Holly and K. T. Foley, "Intraoperative spinal navigation," *Spine,* vol. 28, pp. S54-S61, AUG 1 2003.

[95] A. M. DiGioia, "Computer assisted orthopaedic surgery: Medical robotics and image guided surgery - Comment," *Clinical Orthopaedics and Related Research,* pp. 2-4, SEP 1998.

[96] D. M. Cash, T. K. Sinha, W. C. Chapman, H. Terawaki, B. M. Dawant, R. L. Galloway*, et al.*, "Incorporation of a laser range scanner into image-guided liver surgery: Surface acquisition, registration, and tracking," *Medical Physics,* vol. 30, pp. 1671-1682, JUL 2003.

[97] A. Pandya and L. Zamorano, "Augmented Tracking Using Video, Computer Data and/or Sensing Technologies," US Patent application 20030179308, 2002.

[98] A. K. Pandya, M. Siadat, Z. Ye, M. Prasad, G. Auner, L. Zamorano*, et al.*, "Medical Robot Vision Augmentation—A Prototype," in *Medicine Meets Virtual Reality*, Newport Beach, California, 2003, p. 85.

[99] A. K. Pandya, M. Siadat, L. Zamorano, J. Gong, Q. Li, J. C. Maida*, et al.*, "Augmented Robotics for Neurosurgery," in *American Association of Neurological Surgeons*, Toronto, Ontario, 2001.

[100] A. K. Pandya, L. Zamorano, M. Siadat, Q. Li, J. Gong, and J. C. Maida, "Augmented Robotics for Medical and Space Applications," in *Human Systems*, NASA Johnson Space Center, Houston, TX, 2001.

[101] A. Pandya and G. Auner, "Simultaneous Augmented and Virtual Reality for Surgical Navigation," in *North American Fuzzy Information Processing Society Annual Conference*, Ann Arbor, Michigan, 2005, pp. 429-435.

[102] E. Samset and H. Hirschberg, "Image-guided stereotaxy in the interventional MRI," *Minimally Invasive Neurosurgery,* vol. 46, pp. 5-10, FEB 2003.

[103] N. Nakao, K. Nakai, and T. Itakura, "Updating of neuronavigation based on images intraoperatively acquired with a mobile computerized tomographic scanner: Technical note," *Minimally Invasive Neurosurgery,* vol. 46, pp. 117-120, Apr 2003.

[104] B. King, "Optimal port placement and automated robotic positioning for instrumented laparoscopic biosensors," Ph.D. Thesis, Computer Engineering, Wayne State University Dissertations, 2011.

[105] B. W. Mooring, Z. S. Roth, and M. R. Driels, *Fundamentals of Manipulator Calibration*, 1st ed. New York, NY: John Wiley & Sons, 1991.

[106] M. Gniadecka, P. A. Philipsen, S. Sigurdsson, S. Wessel, O. F. Nielsen, D. H. Christensen*, et al.*, "Melanoma Diagnosis by Raman Spectroscopy and Neural Networks: Structure Alterations in Proteins and Lipids in Intact Cancer Tissue," *Journal of Investigative Dermatology,* vol. 122, pp. 443-449, 2004.

[107] A. R. de Paula Jr and S. Sathaiah, "Raman spectroscopy for diagnosis of atherosclerosis: a rapid analysis using neural networks," *Med Eng Phys,* vol. 27, pp. 237-44, 2005.

[108] R. H. Taylor and D. Stoianovici, "Medical Robotics in Computer-Integrated Surgery," *IEEE Transactions on Robotics and Automation,* vol. 19, pp. 765-782, October 2003.

[109] J. M. Hollerbach and C. W. Wampler, "The Calibration Index and Taxonomy for Robot Kinematic Calibration Methods," *The International Journal of Robotics Research,* vol. 15, pp. 573-591, 1996.

[110] D. J. Bennett and J. M. Hollerbach, "Autonomous calibration of single-loop closed kinematic chains formed by manipulators with passive endpoint constraints," *IEEE Transactions on Robotics and Automation,* vol. 7, pp. 597-606, 1991.

[111] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Heidelberg: Springer, 2008.

[112] W. Khalil, M. Gautier, and C. Enguehard, "Identifiable parameters and optimum configurations for robots calibration," *Robotica,* vol. 9, pp. 63-70, 1991.

[113] L. Giugovaz and J. M. Hollerbach, "Closed-loop kinematic calibration of the Sarcos Dextrous Arm," in *Intelligent Robots and Systems '94*, 1994, pp. 329-334.

[114] H. Zhuang, S. H. Motaghedi, and Z. S. Roth, "Robot calibration with planar constraints," in *Robotics and Automation*, 1999, pp. 805-810.

[115] M. Ikits and J. M. Hollerbach, "Kinematic calibration using a plane constraint," in *Robotics and Automation*, 1997, pp. 3191-3196.

[116] Y. J. Chiu and M. H. Perng, "Self-calibration of a general hexapod manipulator using cylinder constraints," *International Journal of Machine Tools and Manufacture,* vol. 43, pp. 1051-1066, 2003.

[117] A. Nahvi, J. M. Hollerbach, and V. Hayward, "Calibration of a parallel robot using multiple kinematic closed loops," in *Robotics and Automation*, 1994, pp. 407-412.

[118] H. Zhuang, "Self-calibration of parallel mechanisms with a case study on Stewart platforms," *IEEE Transactions on Robotics and Automation,* vol. 13, pp. 387-397, 1997.

[119] J. M. Hollerbach and D. M. Lokhorst, "Closed-loop kinematic calibration of the RSI 6-DOF hand controller," *IEEE Transactions on Robotics and Automation,* vol. 11, pp. 352-359, 1995.

[120] P. G. de Santos, M. A. Jiménez, and M. A. Armada, "Improving the motion of walking machines by autonomous kinematic calibration," *Autonomous Robots,* vol. 12, pp. 187-199, 2002.

[121] D. J. Bennett, D. Geiger, and J. M. Hollerbach, "Autonomous robot calibration for hand-eye coordination," *The International journal of robotics research,* vol. 10, pp. 550-559, 1991.

[122] C. W. Wampler, J. M. Hollerbach, and T. Arai, "An implicit loop method for kinematic calibration and its application to closed-chain mechanisms," *IEEE Transactions on Robotics and Automation,* vol. 11, pp. 710-724, 1995.

[123] Y. J. Chiu and M. H. Perng, "Self-calibration of a general hexapod manipulator with enhanced precision in 5-DOF motions," *Mechanism and Machine Theory,* vol. 39, pp. 1-23, 2004.

[124] Y. Meng and H. Zhuang, "Autonomous robot calibration using vision technology," *Robotics and Computer-Integrated Manufacturing,* vol. 23, pp. 436-446, 2007.

[125] G. Yang, I. M. Chen, S. H. Yeo, and W. K. Lim, "Simultaneous base and tool calibration for self-calibrated parallel robots," *Robotica,* vol. 20, pp. 367-374, 2002.

[126] J. Craig, "Calibration of industrial robots," in *Proceedings of the International Symposium on Industrial Robots*, 1993, pp. 889-893.

[127] M. A. Meggiolaro, G. Scriffignano, and S. Dubowsky, "Analysis and Optimization of Closed-Loop Manipulator Calibration with Fixed Endpoint," in *Proceedings of the CONEM*, Brazil, 2002.

[128] M. A. Meggiolaro, G. Scriffignano, and S. Dubowsky, "Manipulator calibration using a single endpoint contact constraint," in *Proceedings of the DETC2000*, Baltimore, MD, 2000, pp. 1-9.

[129] T. Haidegger, T. Xia, and P. Kazanzides, "Accuracy improvement of a neurosurgical robot system," in *Biomedical Robotics and Biomechatronics*, 2008, pp. 836-841.

[130] M. Abtahi, H. Pendar, A. Alasty, and G. R. Vossoughi, "Calibration of parallel kinematic machine tools using mobility constraint on the tool center point," *The International Journal of Advanced Manufacturing Technology,* vol. 45, pp. 531-539, 2009.

[131] C. S. Gatla, R. Lumia, J. Wood, and G. Starr, "An automated method to calibrate industrial robots using a virtual closed kinematic chain," *IEEE Transactions on Robotics,* vol. 23, pp. 1105-1116, 2007.

[132] N. Tan, C. Clévy, G. J. Laurent, and N. Chaillet, "Calibration and validation of XYΘ micropositioners with vision," in *Advanced Intelligent Mechatronics*, 2012, pp. 256-261.

[133] P. Kazanzides, J. Zuhars, B. Mittelstadt, and R. H. Taylor, "Force sensing and control for a surgical robot," in *Robotics and Automation*, 1992, pp. 612-617.

[134] S. Hayati and M. Mirmirani, "Improving the absolute positioning accuracy of robot manipulators," *Journal of Robotic Systems,* vol. 2, pp. 397-413, 1985.

[135] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*, 1st ed. Springer, 2011.

[136] P. I. Corke and B. Armstrong-Helouvry, "A search for consensus among model parameters reported for the PUMA 560 robot," in *Robotics and Automation*, 1994, pp. 1608-1613.

[137] Y. S. Kwoh, J. Hou, E. A. Jonckheere, and S. Hayati, "A robot with improved absolute positioning accuracy for CT guided stereotactic brain surgery," *IEEE Transactions on Biomedical Engineering,* vol. 35, pp. 153-160, 1988.

[138] R. P. Paul and H. Zhang, "Computationally efficient kinematics for manipulators with spherical wrists based on the homogeneous transformation representation," *The International journal of robotics research,* vol. 5, pp. 32-44, 1986.

[139] S. Chiaverini, L. Sciavicco, and B. Siciliano, "Control of robotic systems through singularities," *Advanced Robot Control,* pp. 285-295, 1991.

ABSTRACT

TECHNIQUES FOR SENSOR-INTEGRATED ROBOTIC SYSTEMS: RAMAN SPECTRA ANALYSIS, IMAGE GUIDANCE, AND KINEMATIC CALIBRATION

by

LUKE ANTHONY REISNER

December 2012

Advisor: Dr. Abhilash Pandya

Major: Computer Engineering

Degree: Doctor of Philosophy

Robotics and sensor technology have made impressive advancements over the years. There are now robotic systems that help perform surgeries or explore the surface of Mars, and there are sensors that detect trace amounts of explosives or identify diseased human tissue. The most powerful systems integrate robots and sensors, which are natural complements to each other. Sensors can provide information that might otherwise be unavailable due to indirect robotic manipulation (e.g., images of the target environment), and robots can provide suitably precise positioning of an analytical sensor.

To have an effective sensor-integrated robotic system, multiple capabilities are needed in the areas of sensors, robotics, and techniques for robot/sensor integration. However, for many types of applications, there are shortcomings in the current technologies employed to provide these capabilities. For the analysis of complex sensor signals, there is a need for improved algorithms and open platforms that enable techniques to be shared. For the path planning and tracking of integrated sensors and the visualization of collected information, image guidance systems that support advanced analytical sensors would be very beneficial. For robotic placement of a sensor,

easily usable calibration procedures and methods to overcome limited feedback could help improve the accuracy.

To help address these issues, some novel systems and techniques were developed in this research. First, a software system was created to process, analyze, and classify data from a specific kind of sensor (a Raman spectrometer). The system is open and extensible, and it contains novel techniques for processing and analyzing the sensor data. Second, an image guidance system was made for use with a sensor-integrated robotic system (a Raman probe attached to a surgical system). The system supports tool tracking, sensor activation, real-time sensor data analysis, and presentation of the results in a 3D computer visualization of the environment. Third, a kinematic calibration technique was developed for serial manipulators. It requires no external measurement devices for calibration, provides solutions for some limitations of existing techniques, and can significantly enhance the positional accuracy of a robot to improve sensor placement.

The implemented techniques and systems were successfully evaluated using various data sets and conditions. Together, the contributions of this work provide important building blocks for an accurate robot with an integrated analytical sensor. This type of a system would be a powerful tool for many future applications, such as a surgical robot that automatically scans for diseased tissue and assists the surgeon in the necessary treatment. Ultimately, this work is intended to foster the development of advanced sensor-integrated robotic systems.

AUTOBIOGRAPHICAL STATEMENT

Luke Reisner is a research engineer at Wayne State University. His most recent degree is a Ph.D. in Computer Engineering from Wayne State University. He also holds a M.S. in Computer Engineering, a B.S. in Computer Engineering, and a B.S. in Electrical Engineering from the University of Michigan–Dearborn. His research interests include digital signal processing, machine learning, image-guided surgery, medical robotics, and Raman spectroscopy.