APPLICATION OF THE ISO 9283 STANDARD TO TEST REPEATABILITY OF THE BAXTER ROBOT

BY

KEVIN S. CHEN

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2015

Urbana, Illinois

Adviser:

Associate Professor Timothy Bretl

ABSTRACT

Industrial robots are becoming more cost-efficient and safer to work with. Rethink Robotics has created an industrial robot called Baxter that is less expensive and more compliant. Unfortunately, while Rethink has an accuracy specification for Baxter, there is some confusion about the repeatability specification of Baxter. This thesis applies the ISO 9283 standard to test the manipulation repeatability of Baxter using an IR motion capture system. This test method varies movement speed, payload weight, and target location while measuring repeatability. The average positional repeatability was determined to be 2.9 mm and 3.3 mm for the left and right arms respectively. The average orientation repeatability was determined to be 0.003 to 0.0047 radians for the left arm and 0.0037 to 0.0054 radians for the right arm.

To my parents, for their love and support.

ACKNOWLEDGMENTS

Special thanks to Jacob Wagner for helping with working with Baxter and the motion capture system. Special thanks to Zihao Zhang for the design and prototyping of the custom end effector fixture.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Industrial robots are constantly improving and changing to become more cost efficient and safer. The Baxter robot from Rethink Robotics is one such robot that is designed to work alongside humans safely and be affordable [1]. In order to best utilize Baxter, there needs to be an understanding of its performance. Robotic performance is usually rated by both accuracy and repeatability specifications.

This chapter will provide a brief overview and introduction of robotic kinematics, robotic performance, as well as the robot itself, Baxter.

1.1 Robotic Kinematics

Kinematics refers to the study of motion and its geometry [2]. In this case, kinematics refers to the motion of robots and modeling of their movement. Symbolically, we can represent robotic manipulators as a sequence of links and joints which connect to form a kinematic chain [3]. These joints can either be revolute or prismatic. Revolute joints rotate around an axis of rotation while prismatic joints actuate linearly along an axis of movement. Each of these joints adds a degree of freedom (DOF) to the robot [4]. To control the hand (also known as the end effector) of a robot manipulator freely in three dimensional (3D) space, six DOFs are needed. Three DOFs contribute to positioning and three DOFs orient the end effector. The position and orientation (or pose) of the end effector are all defined in relation to some global coordinate frame. A kinematic chain can be defined mathematically as a number of transformations from joint to joint. Generally, the transformations are defined to follow Denavit-Hartenberg (DH) conventions.

Generally, robotic manipulators lack the ability to sense the pose of their end effector. However, most robots have encoders on their joints to provide

measurements on their joint angles. Using the joint angle measurements, one can calculate the end effector pose. The equations that describe the relationship between the pose and the joint angles are known as forward kinematics [3].

In many cases, the joint angles needed to command the robot end effector to achieve some desired pose are unknown. The problem of obtaining the unknown joint angles is known as inverse kinematics [3]. Inverse kinematics is usually a much more difficult problem than forward kinematics. Due to the nonlinear nature of the equations involved, a closed-form solution does not always exist [4]. Also, since multiple configurations of a robot arm can achieve the same end effector pose, solutions to inverse kinematics are not unique. In fact, infinite solutions can exist especially in the case of redundant manipulators, where the DOF of the robot exceeds the six DOFs required for positioning an object in 3D space.

The Denavit-Hartenberg (DH) convention is the most commonly used representation of transformations between links in a kinematic chain for robotics [3]. Instead of the classic six parameters used to represent transformations in 3D space, the DH convention only uses four parameters (also referred to as DH parameters). This is possible because the DH convention makes certain assumptions about the transformations between coordinate frames. However, for the same reasons, DH parameters cannot be used to represent any arbitrary transformation in 3D space.

The four DH parameters are link length (a), link twist (α), link offset (d), and joint angle (θ) [3]. These parameters describe the following (see Figure 1.1):

- a is the length between coordinate frames on the 2nd frame's X axis
- α is the rotation about the 2nd frame's X axis
- d is the length from the 1st frame's X axis to the 2nd frame's X axis along the first frame's Z axis
- θ is the rotation about the 1st frame's Z axis

In order for DH parameters to correctly represent the transformations needed, there must be some constraints enforced on coordinate frame selection. First, the X axis of the following frame must intersect the current frame's Z axis.

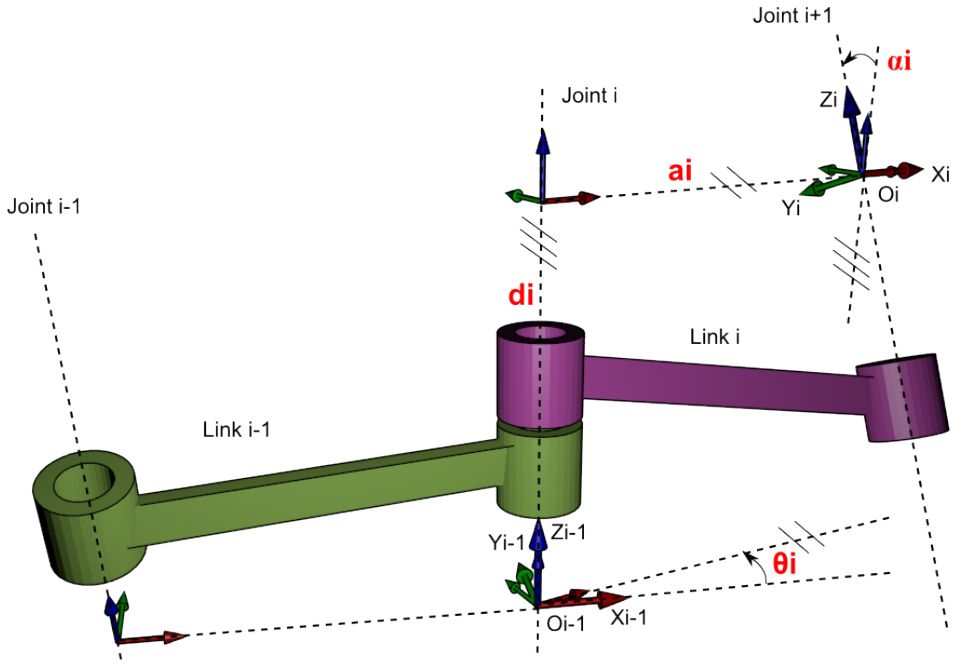

Figure 1.1: DH Parameter Convention [5]

Secondly, the X axis of the following frame must also be perpendicular to the current frame's Z axis. These constraints can clearly be seen in Figure 1.1. Overall, DH parameters combine to form a homogeneous transformation as shown in Equation (1.1). That transformation expands into the form shown in Equation (1.2) where c and s denote cosine and sine of their subscripts, respectively. Of course, since the joints on a robot are actuated, one of the DH parameters is a variable while the other three are constant parameters. In the case of revolute joints, θ is the joint variable. For prismatic joints, d is the joint variable.

$$A_i = Rot_\theta Trans_d Trans_a Rot_\alpha \tag{1.1}$$

$$A_i = \begin{bmatrix} c_\theta & -s_\theta c_\alpha & s_\theta s_\alpha & a c_\theta \\ s_\theta & c_\theta c_\alpha & -c_\theta s_\alpha & a s_\theta \\ 0 & s_\alpha & c_\alpha & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.2}$$

Despite their widespread use in robotic manipulator modeling, DH parameters are not a perfect representation for all robots. The largest issue with the DH convention is that singularities happen when parallel joint axes are

next to each other [6]. The same issue also results in large errors in DH parameters when joints with parallel axes have a small alignment error [7]. A modified DH representation proposed by Hayati is widely adopted to deal with the parallel or near-parallel axes issue. This approach differs from the DH convention in connecting two coordinate frames with a common plane perpendicular to the parent frame's Z axis. Then the parameters are re-defined to θ, a, α, and β, where β represents a rotation about the second frame's Z axis to help align the Z axis with the rotation axis of that joint. The other three parameters retain their DH parameter definitions.

1.2 Robotic Performance

Robotic performance can be described by many specifications including resolution and path control [8]. There are two main metrics of robot manipulator performance: accuracy and repeatability. Accuracy is how close the end effector of a manipulator is to a commanded pose [3]. Repeatability is how close the end effector is when commanded to a previously taught pose. For example, if a robot is commanded to achieve some end effector pose multiple times, the cluster of actual poses achieved would be the repeatability while the difference between the average of that cluster and the original pose would be the accuracy. See Figure 1.2 for a visual example of the difference between accuracy and repeatability.

In general, accuracy and repeatability can be affected by many factors. These factors can include computational errors in the control of the joints or of the mathematical model of the robot's kinematics [8]. They can also be affected by dynamic effects such as warping of links, compliance, or gear backlash. Even manufacturing tolerances can contribute to the accuracy and repeatability of a robot [6].

1.3 Baxter Robot and Motivation

The Baxter robot is developed by Rethink Robotics to be an affordable industrial robot that is compliant and can safely work in a small factory setting alongside human workers [1]. Baxter is a two-armed robot with seven DOFs

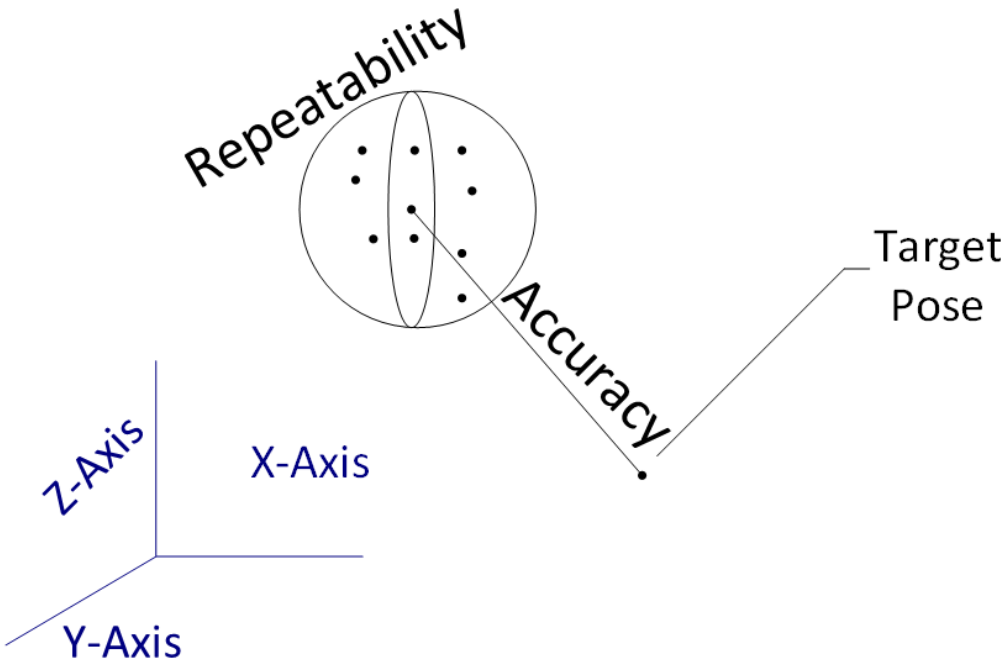

Figure 1.2: Visualization of Accuracy and Repeatability

in each arm (see Figure 1.3). A unique feature of Baxter is that its joints are not directly driven by its motors. Rather, Baxter's joints make use of series elastic actuators (SEA) which consist of having a spring attach the motor to the joint [9]. Traditionally, robotic joints are rigidly attached to a motor either directly or by a drive chain. The SEA configuration allows force and torque measurements by measuring the twist in the spring either by potentiometer or strain gauge. The SEAs also contribute to Baxter's compliance. Baxter also features three cameras (two embedded in the wrist and one in the head), sonar sensors in the head, and infared distance sensors in his wrists [10].

Baxter utilizes the Robot Operating System (ROS) framework alongside Rethink's own Software Development Kit (SDK) for custom programming [11]. The ROS framework is a commonly used middleware for robotic systems. Within ROS, processes, such as robotic systems, form nodes which are connected in a graph network fashion [12]. These nodes communicate via messages published in topics or service requests [13]. Messages are one-way communications where one or multiple nodes publish to subscriber nodes. For two-way communications, services are requested by one node and then

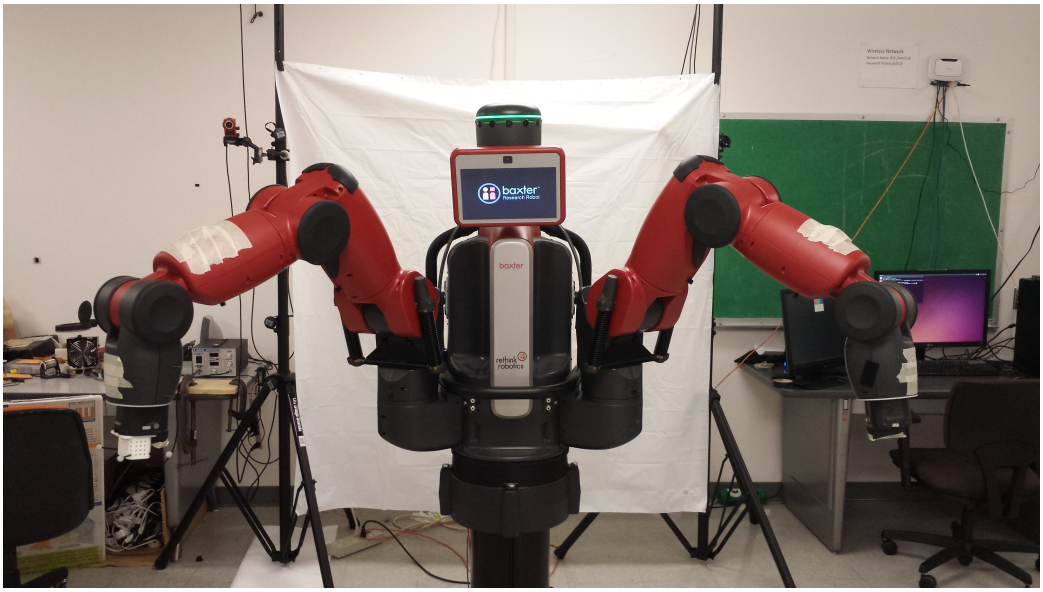

Figure 1.3: Baxter Robot

fulfilled by another node with a reply. With Baxter, its onboard computer has a node which can be commanded through a node setup on a host computer. For example, if the user wanted to command Baxter's limbs to certain joint angles, they would publish joint angles to the joint_command topic [14]. Baxter's node receives the joint angles through that topic and processes the data accordingly. Baxter's execution of the commanded joint angles is abstracted away from the user's node. Baxter constantly publishes joint angle measurements on a joint_states topic which the user can subscribe to through their node to receive the measured joint angles. Baxter also has inverse kinematics functionality as a ROS service.

Baxter's quoted accuracy specification is ± 5 mm [10]. However, Rethink Robotics has not provided a repeatability specification in addition to the accuracy. To better understand the full extent of Baxter's performance, it is important to also know Baxter's repeatability in addition to its accuracy. This thesis aims to characterize Baxter's repeatability through experimentation.

The remainder of this thesis will be split into four chapters. Chapter 2 will review relevant literature. Chapter 3 will describe our approach at characterization of Baxter's repeatability. Chapter 4 will present the results of testing. Chapter 5 will conclude with some final thoughts.

CHAPTER 2

LITERATURE REVIEW

This chapter will present an overview of previous works pertaining to the study and characterization of robotic repeatability.

2.1 Effects of Speed and Payload on Robot Repeatability

Onyebuchi Felix Offodile and Kingsley Ugwu explored the effects of movement speed and payload weight on robotic repeatability in their 1991 paper [15]. They evaluated a PUMA 560 robot by repeatedly commanding the end effector to four different locations while varying speed and weight. While manufacturers provide specifications on robot repeatability, Offodile and Ugwu point out that the conditions under which repeatability was evaluated are not known. They hoped to help users better evaluate robotic systems for specific applications knowing how speed and payload might affect repeatability.

2.1.1 Method

For their experiment, the PUMA 560 robot is set up next to an ALTEK AC-30 digitizing tablet and a stylus is attached as an end effector. The PUMA 560 robot is a six DOF industrial manipulator with a listed repeatability specification of ± 0.1 mm. It has a maximum rated payload of 2.27 kg and maximum velocity of 1016 mm/s. The digitizing tablet is used as the measurement device to record 2D position coordinates at six decimal precision.

Four locations (A-D positioned in an approximately trapezoidal shape with the widest base nearest to the robot) are taught to the robot for each cycle of experimentation. The robot is commanded to each location while varying

its payload and velocity. The payload is varied between 0.45 kg and 1.81 kg at 0.45 kg increments. Speed of movement is also varied from 10%-100% in 10% increments. The target locations were retaught for every weight cycle. This is necessary to circumvent possible misalignment errors due to the user changing the weights on the robot. It should be noted that since the measurement device is a digitizing tablet, all deviations measured are 2D and do not include orientation data.

$$R_i = \sqrt{(X_i - X_o)^2 + (Y_i - Y_o)^2} \tag{2.1}$$

$$R_r = (\sum_{i=1}^{10} R_i)/n \tag{2.2}$$

Repeatability is calculated in this paper using Equations (2.1) and (2.2). Equation (2.1) calculates the repeatability for a specific trial i of the experiment. Only positive roots of R_i are used to simplify analysis. Using R_i for all trials i and Equation (2.2), average repeatability (R_r) is calculated. Repeatability was plotted against speed and load to determine their effect on repeatability.

2.1.2 Results

Offodile and Ugwu looked at the average repeatability compared across different speeds and different weights separately. They also looked at the combined effect of speed and weight on repeatability.

When looking individually at the effects of speed on repeatability, Offodile and Ugwu found that higher speeds generally caused repeatability to deteriorate though not significantly, especially when using a lower payload. Lower weight tests showed seemingly constant repeatability for all speeds tested. The best repeatability measurements were taken when the speed of movement was under 50%. However, effects of speed on repeatability seem to follow no clear pattern.

As with the effects of speed on repeatability, repeatability was found to be worse as payload weight increased. Interestingly, this inverse relationship between payload and repeatability does not hold with no payload attached at lower speeds. Offodile and Ugwu observed that under 60% speed, a 0.45 kg payload actually yielded better repeatability than no payload. They hypoth-

esized this effect was due to overshoot and the lighter arm causing higher speeds. The 0.45 kg weight dampened said overshoot just enough to cause a better repeatability performance.

When individually examined, speed and payload both show the best repeatability to be at 40%-50% speed carrying 0.45 kg or 50% speed carrying 0.91 kg. However, when examining the combined effect of speed and payload, the best repeatability was measured under 20%, 90%, and 100% speeds carrying no weight. These minimal repeatability conditions do not match the results of independent analysis of the effects of speed and payload on repeatability. Due to this and finding many other local minima, Offodile and Ugwu concluded that the effects of weight and speed are not independent of each other. They also found the manufacturer's repeatability specification occurred when the robot carried no payload and moved at 70% speed under the combined analysis. They concluded that the specification was an average value and not particularly useful since it was recorded under the conditions of having no payload, which is not an usual operating condition.

From their work, Offodile and Ugwu concluded that slower speeds and less payload weight generally gave the best repeatability results. However, the two factors cannot be independently analyzed to determine the best repeatability conditions. Ultimately, robots should be evaluated by the user for the specific speed and weight conditions required for their application.

2.2 Effects of Target Location on Robot Repeatability

Raziel Riemer and Yael Edan researched the effects of target location on robot repeatability [16]. Edan's past work used statistical analysis to evaluate repeatability differences between various test conditions [17]. These conditions included different robots of the same model, different velocities, and also different target locations. In that work, Edan found statistical evidence that height and location of a target point affected repeatability, but did not directly look at repeatability compared across different target locations. In this work, Riemer and Edan experimentally determine the effect of target location on repeatability and develop an error-analysis model to predict repeatability based on target location.

2.2.1 Method

A CRS A255 robot with 5 DOF and specified repeatability of 0.05 mm is used for experimentation. It can carry a maximum payload of 2 kg and move with a maximum speed of 0.508 m/s. A three-dimensional measurement system of dial gauges, each with a capacitive Sylvac system, is used for measurement of position. The measurement system has an accuracy of 0.005 mm. Due to the limitations of the measurement system (only linear measurements), orientation is ignored in this research.

Since previous work showed that speed and payload weight had a combined effect on repeatability, Riemer and Edan first determined the speed and payload weights that achieved the best repeatability for their robotic system [15]. This was done via a preliminary experiment where repeatability was measured while payload and speed were varied. Payload was set at 50% and 100% of maximum payload. Speed was set at 50%, 80%, and 100% of the maximum rated speed. They determined the best parameters for their operation were a payload of 2 kg running at 80% of the maximum speed.

To evaluate the effects of target locations on repeatability, nine locations at three heights were chosen as the target locations. Those points were positioned at 10% of the max reachable space of the robot. Due to the large amount of experimentation required, only 15 of the original 27 locations were used (five per height). The original nine positions per height were set in a three by three formation. The new five locations per height are the corner positions of the square and the center of the square. The end effector was commanded to pass through each target point and enter the measuring system at a 45 degree angle. Each experiment consisted of 30 cycles as per ISO 9283 specifications and was run 30 times for a total of 900 measurements per target location.

Statistical methods were used to evaluate differences and compare the repeatability measurements. Friedman's a-parametric analysis was used to determine statistical difference between different target locations. The t-test was utilized to compare identical target locations at different heights.

Previous work by Veitschegger and Wu suggested a theoretical model for predicting accuracy and repeatability based on the kinematic model of a robot [18]. However, Riemer and Edan developed their own theoretical model citing that Vetischegger's model had not been experimentally validated and

was difficult to utilize due to calculation of individual joint error using matrix transforms. Riemer's and Edan's error-analysis model was also derived from the robot's kinematic model similarly to Veitschegger's but was much simpler due to not having to calculate individual first and second order error terms [18]. The error analysis model consists of four steps. First, inverse kinematics is used to determine the joint angles for a desired target location. Partial derivatives of the forward kinematic model are then calculated. Next, the experimental joint angle error and robot angles are substituted into the partial derivatives of the forward kinematics model. From the partial derivatives, statistical and absolute errors are calculated for the total error at some target location. The equations for statistical and absolute error are Equations (2.3) and (2.4) respectively.

$$E_r = \sqrt{E_x^2 + E_y^2 + E_z^2} \tag{2.3}$$

$$E_r = \sqrt{|E_x| + |E_y| + |E_z|} \tag{2.4}$$

2.2.2 Results

Repeatability is calculated differently than in Offodile and Ugwu's work. Instead, Riemer and Edan follow the ISO9283 standard of calculating repeatability using the average deviation of position in 3D space and the standard deviation of the average deviation.

Riemer and Edan applied Friedman's test to determine statistical difference between repeatability at different target locations. They found that different target locations' repeatability had significant differences. There were also significant differences in the repeatability of relative positions, different locations at the same height, and the same location at different heights.

The experimental repeatability was compared against the estimated repeatability of both the statistical and absolute error models. They verified that their statistical error model had similar results to the experimental results. Using the statistical error model, a 3D mapping of repeatability at different heights was generated against the X and Y positioning of target locations. This 3D mapping shows that repeatability generally gets worse the farther away the target location is from the robot.

2.3 ISO 9283

ISO 9283 is the international standard for industrial robot performance criteria and test methods [19]. When ISO 9283 was first defined, less than 10% of companies actually employed it for testing [20]. However, studies found manufacturers and users both considered performance testing necessary and users wanted testing more in terms of final applications [20]. ISO 9283 includes methods for testing many performance characteristics including stabilization time, position overshoot, static compliance, etc. Specifically, the focus of this section will be on the methods of testing for pose repeatability.

2.3.1 Importance of Standardization

ISO 9283 plays an important role in the standardization of test methods and test conditions. Robots that are not evaluated using ISO 9283 methods vary in their test methods and conditions [21, 22, 23]. Mooring's repeatability analysis used one target location for testing and approached it from various different directions for 750 cycles [23]. No speed of movement or payload is specified. Mooring did acknowledge the effect of target location on repeatability and provided a method to determine where to have a measurement device placed for repeatability evaluation. Preising's analysis also utilized only one target location which was approached from 10 different starting poses for 500 cycles at 30% of the robot's maximum speed with a camera as a payload and measurement device [22]. Shing's experiment controlled the robot to fifteen target poses spread throughout the SCARA robot's workspace envelope [21]. Each target pose was approached at three different speeds and with three different payloads. For each combination of pose, speed, and payload, thirty measurements were taken. Out of the three, only Shing's experiment varied location, speed, and payload, all of which Offodile and Riemer showed affect repeatability. ISO 9283 helps to ensure that repeatability is evaluated over these parameters and for the same values.

ISO 9283 also helps standardize the calculation of repeatability. Preising and Mooring multiply the inverse of the measured pose's homogenous transformation by the commanded pose's homogenous transformation to find the deviations in position and orientation and then apply the Euclidean norm to the various components [22, 23]. Shing instead calculated repeatability as

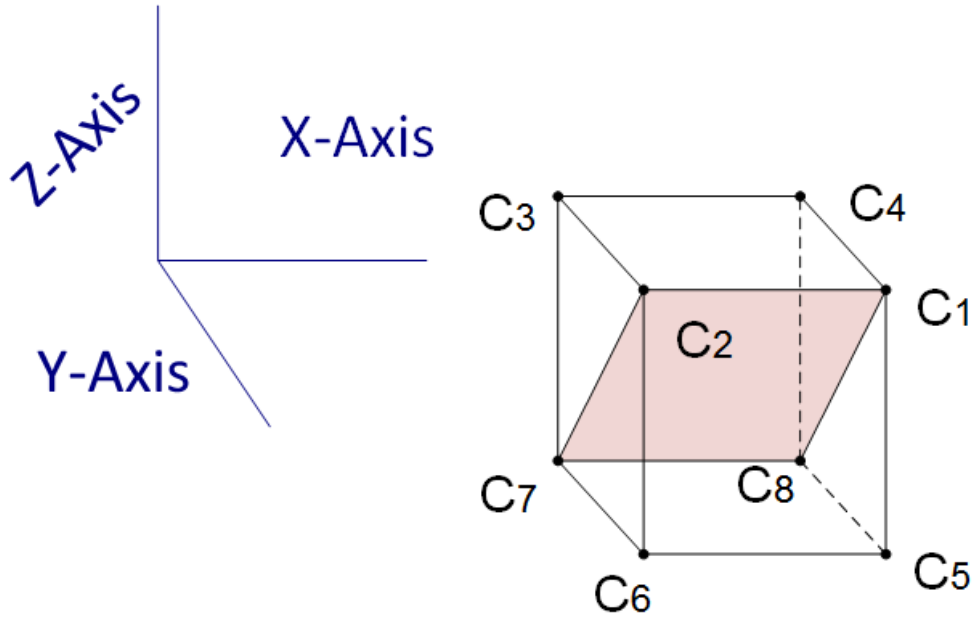

Figure 2.1: Testing Cube Configuration; C_1-C_2-C_7-C_8 Plane

the Euclidean norm of the standard deviation of the measurements in each coordinate frame axis [21]. ISO 9283 makes the calculation of repeatability consistent and therefore a comparable metric across characterization of different robots.

2.3.2 Method

ISO 9283 defines poses used in repeatability tests in relation to a test cube defined in the workspace of the robot. The corners of the cube are designated as C_1 through C_8 (see Figure 2.1). Diagonal pose testing planes are defined in the test cube in relation to the corners. The test plane shown in Figure 2.1 is C_1-C_2-C_7-C_8. The other possible test planes are C_2-C_3-C_8-C_5, C_3-C_4-C_5-C_6, and C_4-C_1-C_6-C_7. There are two requirements to where the cube is placed in the workspace. First, the cube should be in a portion of the workspace that is anticipated to see the most use. Also, the cube should have the largest possible volume with its edges parallel to the base coordinate system of the robot. By following these two requirements, the repeatability specification is more useful because it represents the repeatability for the largest and most used part of the robot's workspace.

On the test plane, five test poses to be used are defined in relation to the

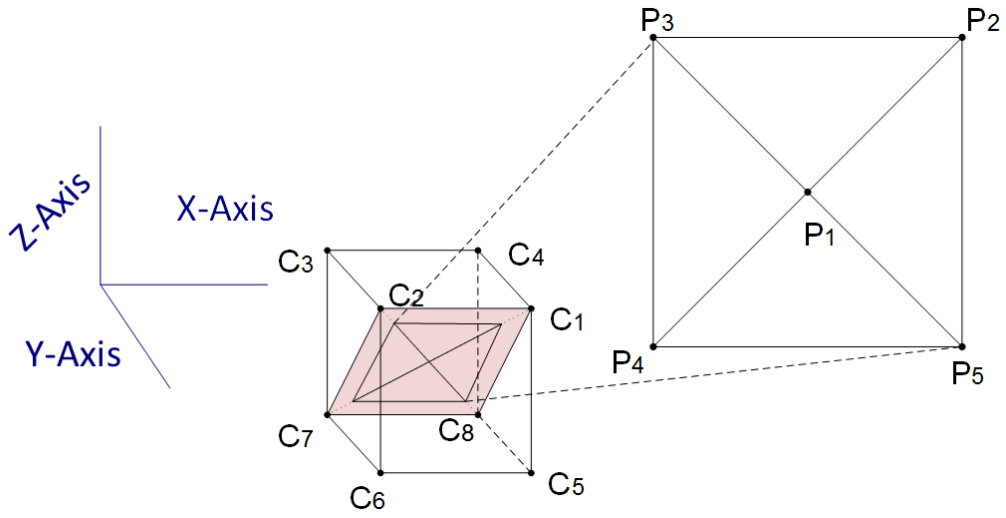

Figure 2.2: Test Poses for Test Cube Shown in Figure 2.1

test cube (P_1 through P_5). P_1 is defined as the center of the cube and the intersection of the diagonals on the test plane. Poses P_2 through P_5 lie on the diagonals of the test plane in the test cube as shown in Figure 2.2. These poses are placed a tenth of the diagonal length of the test cube away from the corners of the diagonal plane. So for the test poses shown in Figure 2.2, P_2 would be placed a tenth of the length of diagonal $C_1 - C_7$ away from C_1. Orientation of the poses is not specified but is left up to the manufacturer. Since the test plane is where the robot wrist is commanded to, the actual measurements are transformed by an axial and radial offset and lie on a measurement plane parallel to the test plane.

The testing procedure consists of commanding the robot to achieve the test poses P_1 through P_5 and recording the actual pose measured. The test poses can be visited in order of either P_1 through P_5 or P_5 through P_1. As long as the order of visitation is consistent in testing and unidirectional, either of the cycle orders can be chosen. Velocity is varied between 10%, 50%, and 100% of manufacturer rated velocity. Payload weight is set at 100% of the rated load and optionally at 10% of the rated load. All combinations of velocity and payload weight should be tested by cycling through the test poses. Each trial of testing is visiting the test poses for 30 cycles. The entire step-by-step procedure is shown in Algorithm 1.

In ISO 9283, repeatability is split into position repeatability and orientation repeatability. Position repeatability is defined as the radius (RP_l) of the

Algorithm 1: ISO 9283 Measurement Procedure for Repeatability Evaluation

```
for Payload ← 100% and 10% do
    for Velocity ← 100%, 50%, and 10% do
        for c ← 1 to 30 do
            for EndEffectorCommand ← P1 to P5 do
                Measure End Effector Pose;
```

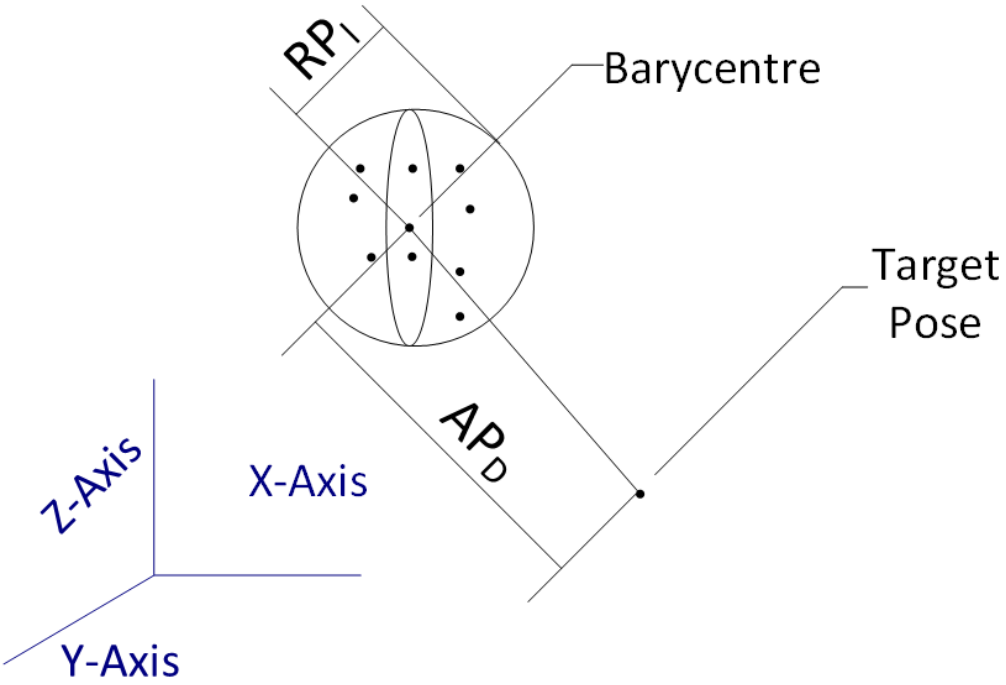

Figure 2.3: Position Repeatability

sphere whose center is the barycentre as shown in Figure 2.3. The barycentre is the point with the mean coordinates ($\bar{x}$, $\bar{y}$, $\bar{z}$) of the cluster of measured points. These mean coordinates are calculated using Equation (2.5) where n is the number of measured points and x_j is the jth measured point. All of the 30 measurements of a singular trial are compared against this barycentre. Equation (2.6) calculates the distance between the jth measured point and the barycentre. Using the distances calculated with Equation (2.6), Equation (2.5) gives the mean distance $\bar{l}$. The mean distance $\bar{l}$ and the distance between each measurement and the barycentre are used in Equation (2.7) to calculate the standard deviation of the distances. ISO 9283's position repeatability specification is then based on the mean and standard deviation of the distances. Position repeatability is calculated using Equation (2.8).

$$\bar{x} = \frac{1}{n}\sum_{j=1}^{n} R_j \tag{2.5}$$

$$l_j = \sqrt{(x_j - \bar{x})^2 + (y_j - \bar{y})^2 + (z_j - \bar{z})^2} \tag{2.6}$$

$$S_l = \sqrt{\frac{\sum_{j=1}^{n}(l_j - \bar{l})^2}{n-1}} \tag{2.7}$$

$$RP_l = \bar{l} + 3S_l \tag{2.8}$$

Orientation repeatability is further split into three separate measures of repeatability (RP_a, RP_b, RP_c). Each measure of orientation repeatability is the repeatability of an angle of rotation and is calculated separately from the others. These repeatability measures are based on the standard deviation of the difference between achieved rotations and the mean rotation. The mean rotation angle is calculated with the same equation as position, Equation (2.5). The standard deviation of differences between the achieved angle and the mean angle is calculated using Equation (2.7). The repeatability specification is essentially three times the standard deviation. The repeatability of each angle of rotation can be calculated using Equation (2.9) where a_j is the jth recorded orientation angle a and $\bar{a}$ is the mean orientation angle. The same equations apply for rotation angles b and c.

$$RP_a = \pm 3S_a = \pm 3\sqrt{\frac{\sum j = 1^n (a_j - \bar{a})^2}{n-1}} \tag{2.9}$$

2.3.3 Other Methods

ISO 9283 is not the only method used to determine repeatability that is currently widely used. Besides ISO 9283, there are methods that incorporate neural networks to determine repeatability or modeling based on the robot's kinematics [24]. While being highly accurate, methods utilizing neural networks are complex and require a large amount of time and data due to the nature of training a neural network. Modeling of a robot's repeatability using its kinematics enables predicting the repeatability at various locations as demonstrated in Riemer and Edan's work [16]. However, certain assumptions are necessary for the model to return accurate results such as the error distribution. If those assumptions are incorrect, then the model will produce incorrect results. Another competing method of repeatability characterization lies in the ANSI/RIA standard 15.05 which will be discussed in Section 2.4.

Another method of determining repeatability is the stochastic ellipsoid method. This method utilizes experimental data and robot kinematics to build a stochastic model of repeatability throughout the workplace [25]. The experimental side of the stochastic ellipsoid method involves measuring the individual variances of each joint of the robot. The measured variances are used to estimate a covariance matrix of angular position. The covariance of angular position in conjunction with the manipulator jacobian are used to obtain the covariance of workspace error. A density function of the position is then formed from the covariance of the workspace and represents the reference stochastic ellipsoid. For a given position and risk, a stochastic ellipsoid can be generated from the reference ellipsoid using a central homothety. The risk represents the probability that a point will lie outside the ellipsoid. Using this method, the repeatability of a robot can be generated for the entire workspace.

While the stochastic ellipsoid method is able to generate repeatability predictions for the entire workspace using less experimentation, it is difficult to implement in reality. The main problem is the difficulty of the experimenta-

tion. Variability needs to be measured for each individual joint without the influence of the others. Brethe et al. attribute their success at measuring individual joint variability to strong brakes on the joints and a low operation speed [26]. Also the robots involved in experimentation were generally lower DOF robots. Applying the stochastic ellipsoid method to a higher DOF robot will be difficult.

2.4 ANSI/RIA R15.05

ANSI/RIA R15.05 is the rival standard to the ISO 9283 standard for robotic performance evaluation. The ANSI standard mainly deals with the evaluation of positional accuracy and positional repeatability. Orientation errors are mostly ignored given that any orientation errors would be indirectly measured through positional deviations [27]. Some of the other characteristics evaluated include overshoot, settling time, and compliance. This section will focus on the repeatability evaluation method provided in ANSI/RIA R15.05.

2.4.1 Method

Instead of defining the test poses in relation to a test cube like in ISO 9283, test poses in ANSI/RIA R15.05 are defined in terms of a test plane and a test path. As seen in Figure 2.4, the left plane is the standard test plane while the right plane is the exception test plane. The general test plane is any plane parallel to a (1,1,-1) plane as defined in the robot's coordinate frame and passes through the work space center point. All robots that are able to achieve six DOF poses should use the general test planes. Robots with less that six DOFs should use the exception test plane which is parallel to a (1,1,0) plane. The test path which lies on the test plane is shown in Figure 2.5. The test path is divided into n segments of S_L length which can be either 200, 500, or 1000 mm. The D_L segment side length is defined to be half of S_L. Test points L_1, $L_{(n+1)/2}$, and L_{n+1} are the test points used in repeatability testing. Test points series L and U are used together for other tests. The center line that connects F_1 and F_2 in the test path intersects the reference coordinate frame axes at points E_1 and E_2. The largest segment length possible should be selected such that the test path has at least three

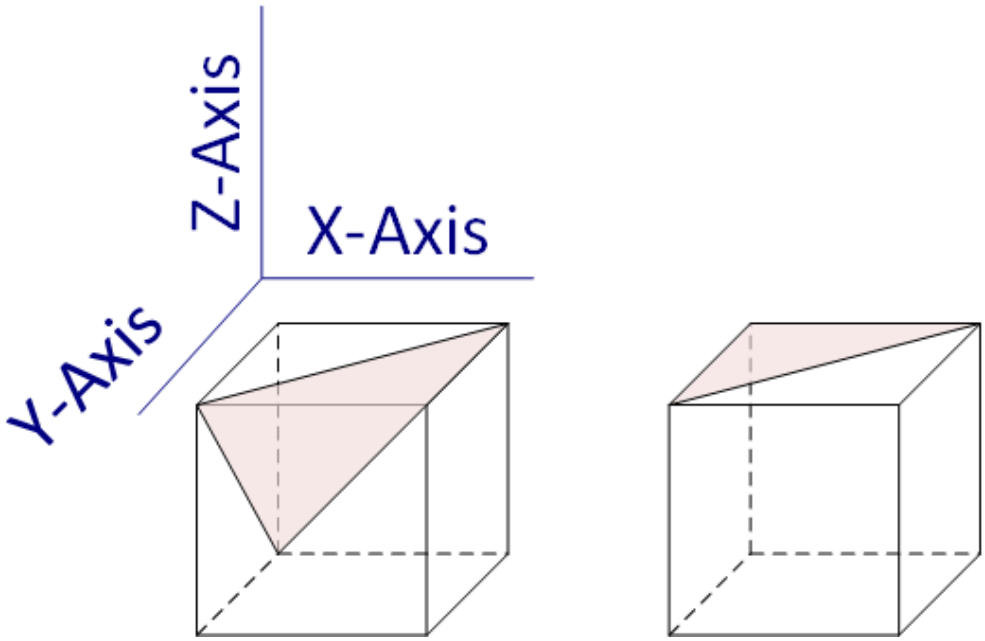

Figure 2.4: ANSI/RIA R15.05 Test Planes

segments. The orientation of the pose at each test point is only specified to have the z axis of the end effector perpendicular to the test plane. The z axis is defined to be perpendicular to the mechanical flange of the robot and has its origin at the center point of the flange.

Testing is done by commanding the robot end effector to three measurement points L_1, $L_{(n+1)/2}$, and L_{n+1} along the test path. This is done for at least 500 cycles for N measurements and measured for the deviations from the mean position at each point. The test payload is selected as close as possible to the manufacturer's maximum payload rating between 12 different load categories. The load categories are 1 kg, 2 kg, 5 kg, and 10 kg to 140 kg with 20 kg increments. The 12th load category is unspecified and for robots with rated payloads over 140 kg. No velocity of movement is specified.

Repeatability is calculated similarly to ISO 9283 specifications. A mean centroid is computed for each measurement point using Equation (2.5). The mean repeatability is the average distance of each measurement from the centroid points using Equation (2.10) where r_{ai} through r_{ci} are calculated using Equation (2.6). The standard deviation of the repeatability is also calculated from the deviation of each measurement and the mean repeatability using Equation (2.11). Both $\bar{r}_{REP}$ and S_{REP} are used as the repeatability performance. Equation (2.8) is only used if the data is found to be normally

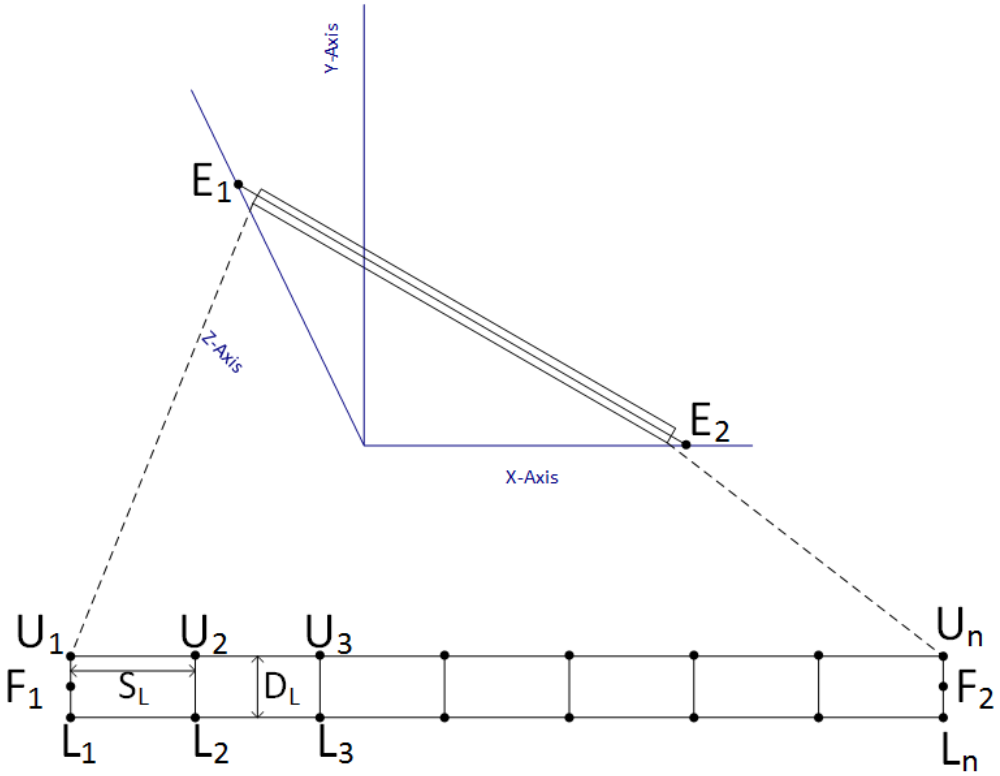

Figure 2.5: ANSI/RIA R15.05 Test Path

distributed. It is used to represent the radius of a sphere that would bound 99.7% of the results.

$$\bar{r}_{REP} = \frac{\sum_{i=1}^{N} r_{ai} + \sum_{i=1}^{N} r_{bi} + \sum_{i=1}^{N} r_{ci}}{3N} \tag{2.10}$$

$$S_{REP} = \sqrt{\frac{\sum_{i=1}^{N}(r_{ai} - \bar{r}_{REP})^2 + \sum_{i=1}^{N}(r_{bi} - \bar{r}_{REP})^2 + \sum_{i=1}^{N}(r_{ci} - \bar{r}_{REP})^2}{3N - 1}} \tag{2.11}$$

2.4.2 Differences from ISO 9283

There are many differences between the ISO 9283 standard and ANSI/RIA R15.05 standard. Differences in repeatability determination will be focused on in this section. Poses are specified differently between the two standards. ISO 9283 requires five poses arranged in a square for repeatability testing while ANSI requires only three poses along a rectangular plane. Orientation is only specified by ANSI, but as Jeswiet and Helferty point out, the roll component of the orientation is not specified [28]. ANSI ignores orientation repeatability while ISO separates it as its own specification. ISO also requires much less data collection running only 30 cycles of testing, while ANSI requires 500 cycles. Jeswiet and Helferty demonstrated that the 30 cycles suggested by ISO was insufficient to determine long-term repeatability and at least 200 samples may be necessary

Another major difference is the consideration of payload and movement speed. ANSI specifies the selection of one of 12 weight categories. ISO 9283 specifies always testing with 100% of rated payload and optionally with 10% rated payload. ISO 9283 also specifies testing with 100%, 50%, and 10% of the maximum velocity of the robot while ANSI makes no specification about the movement speed of the robot. This is an important difference since previous work has made it clear that payload and speed do have significant effects on repeatability [15].

Jeswiet and Helferty point out some shortcomings of both the ISO and ANSI standards. They explored some preliminary effects of uncertainty on the final repeatability measurements [28]. They also looked at the effects

of measurement equipment on the repeatability. Using their results, they suggested possible improvements to the standards including the requirement of determining uncertainty and of the measurement equipment.

CHAPTER 3

METHOD

This chapter will detail the experimental process used to determine the repeatability of Baxter as well as the preliminary work done prior to experimentation. The process will be split into the hardware setup, the measurement procedure, and the data analysis.

3.1 Hardware Setup

The hardware setup consisted mainly of the Baxter robot and a motion capture system as the measurement device (see Figure 3.1). The Baxter robot was controlled over LAN network by a computer running Ubuntu 12.04 as its operating system. The motion capture system was interfaced over USB 2.0 with a Windows 8.1 computer. Figure 3.2 shows the block diagram of the described hardware setup.

3.1.1 Baxter

This subsection will describe the details of working with the Baxter robot. This will include an overview of Baxter's internal processing, the results of preliminary testing done with Baxter, and the various considerations when implementing the repeatability testing of Baxter.

3.1.1.1 Overview of Baxter's Internal Processes

Baxter is controlled through a LAN by a Linux desktop computer. It is interfaced through the ROS framework. ROS can be used by both C++ or Python language due to its node network organization [12]. The desktop commands Baxter using ROS messages published to ROS topics that Baxter

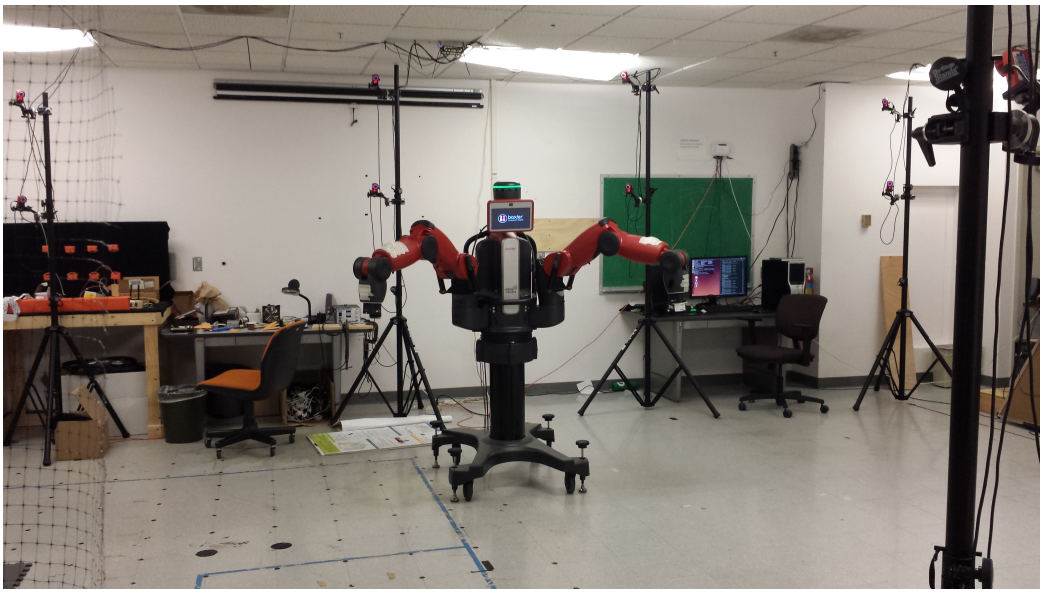

Figure 3.1: Experiment Hardware Setup

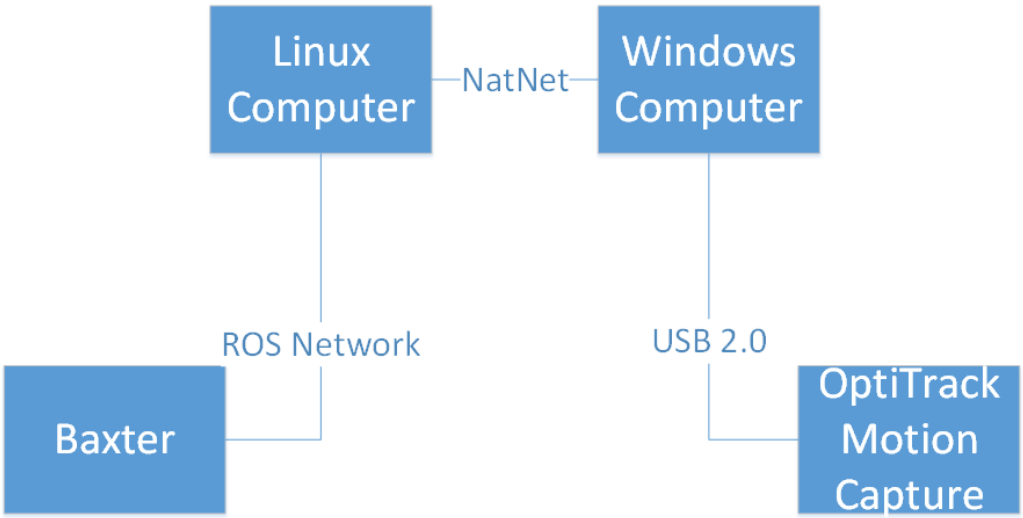

Figure 3.2: Block Diagram of Hardware Setup

is listening to at 100 Hz. Baxter's internal Linux computer receives the published messages over LAN via joint control listeners [29]. Separate from the joint control listeners, there is a control loop running at 1kHz that handles all control over Baxter's joints as well as recording of sensory data from Baxter's sensors. This control loop is run at the highest priority level within Baxter's internal Linux PC to ensure the 1 kHz rate of operation. The control loop will asynchronously check the control listeners for new ROS messages to process. The new commands are then used to control the individual joint control boards on each of Baxter's joints. This entire process is shown in Figure 3.3.

Due to the number of intermediary processes between the user program on the user workstation and the actuation of Baxter's joints, there is a large amount of delay in the system. From when the user program publishes a message for Baxter to when Baxter's joint control listeners receive it, there is approximately 1.6 ms of delay [29]. There may be even more delay in this step due to network conditions. Internally, Baxter also has delay between message reception and command execution. The control loop takes approximately 1 ms to receive the command from the control listener and then another 1 ms to command the joint control boards. In total from user program command to Baxter's joints executing the command, there is about 3.6 ms of delay. The same delays also exist for data from Baxter's joint encoders back to the user program. So the total delay from user command to first being able to detect the execution of the command via joint feedback is about 7.2 ms. This delay loop is visualized in Figure 3.4. It must also be noted that Baxter's internal PC is completely protected and does not allow any modification of its internal systems.

3.1.1.2 Preliminary Testing

In order to obtain a better understanding of the Baxter robot, some preliminary testing was done. A simple preliminary test was run to see the effects of the delays in Baxter's system. Baxter was commanded to follow a 2D sinusoidal path vertically (z axis). The commanded z axis position data and measured z axis position data were plotted against time. The plots were also generated for velocity data. As seen in Figure 3.5, the roundtrip system delay is noticeable but seems to be inconsistent. From the velocity

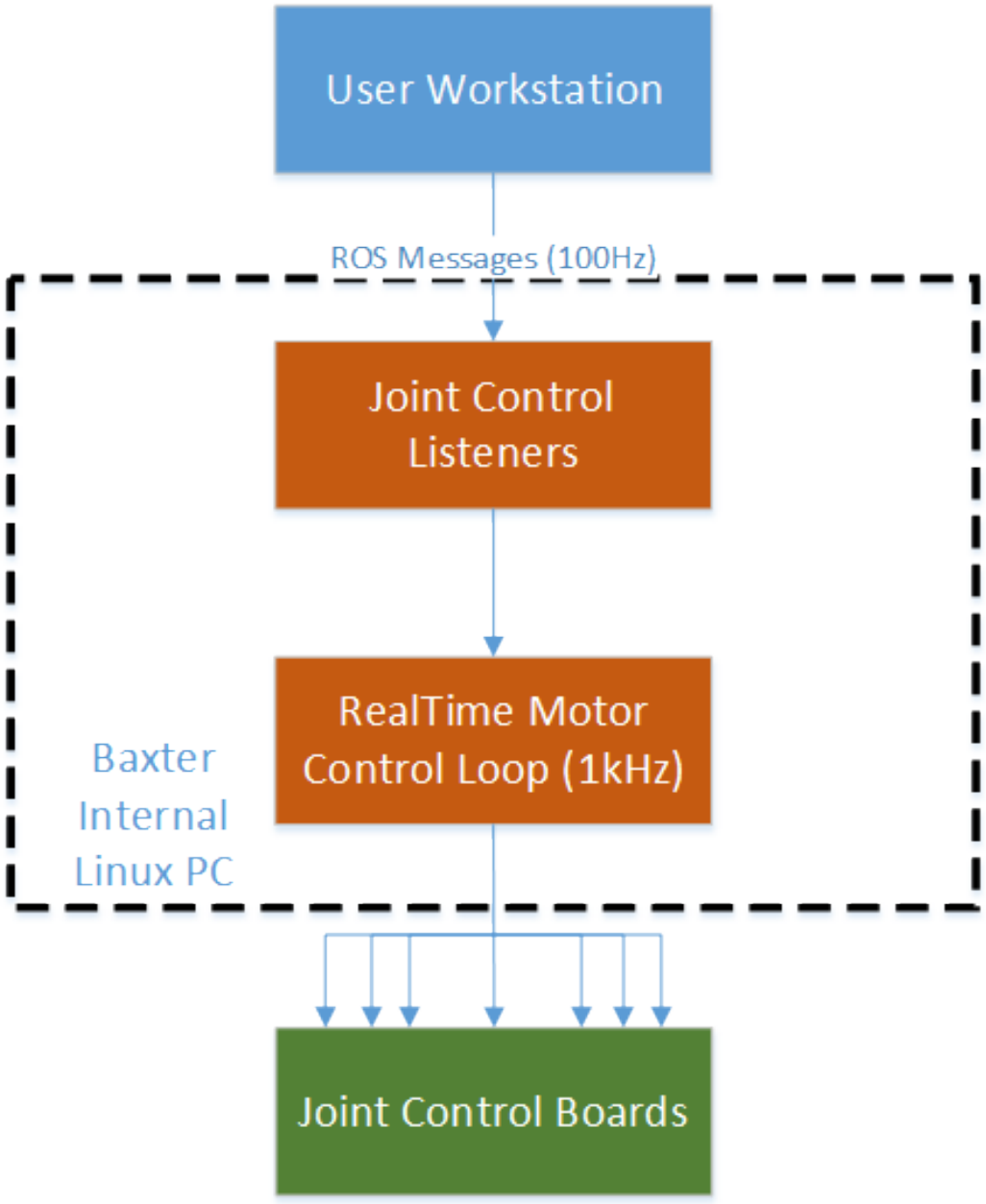

Figure 3.3: Baxter's Internal Hardware

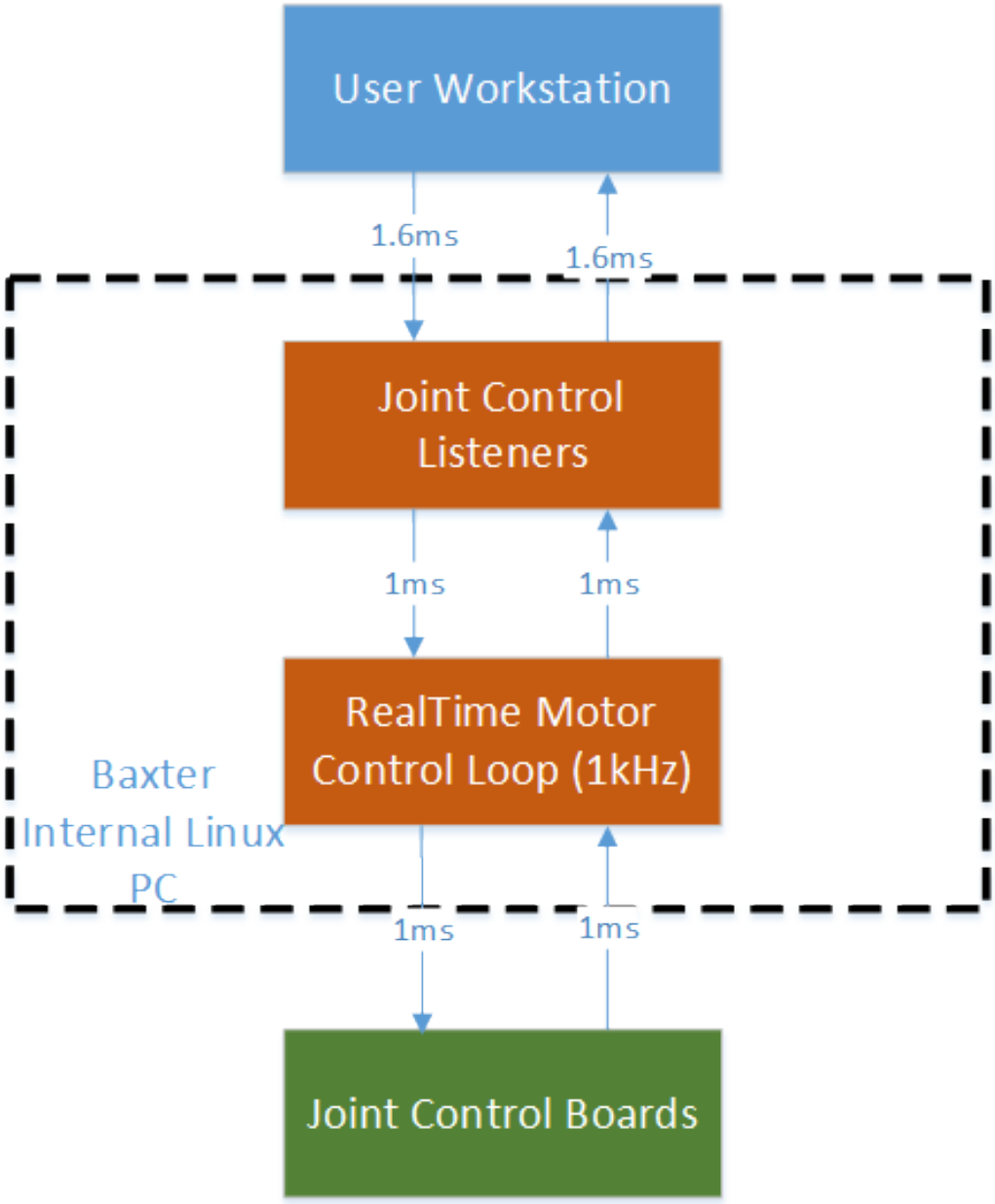

Figure 3.4: Baxter's Internal Hardware Timings

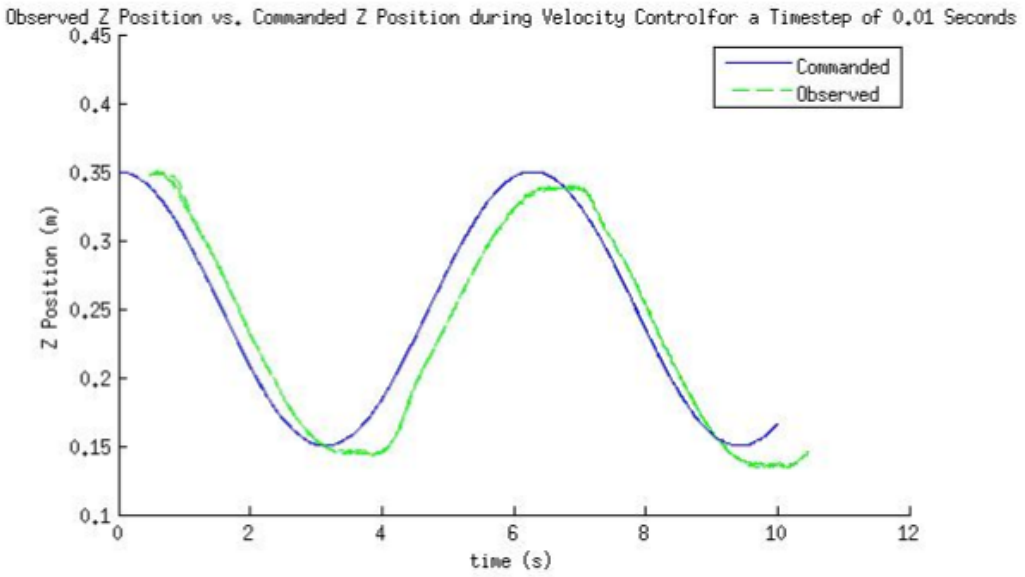

Figure 3.5: Commanded and Recorded End Effector Z Position Data

plot in Figure 3.6, one can see a perturbation whenever the observed velocity becomes zero. This suggests that Baxter's joints may be demonstrating stick-slip phenomenon. It should be noted that the errors in positional tracking cannot be taken as a representation of Baxter's positional accuracy since the end effector pose is calculated internally from joint measurements and Baxter's internal forward kinematics model. Therefore errors in Baxter's internal model may contribute to the perceived errors in Figures 3.5 and 3.6.

From these preliminary tests, some unusual behavior with the recording of data was observed. It was observed that some data samples would disappear or multiple samples would be delayed and come together in bursts. It was hypothesized that this abnormal behavior was either due to the Python language or due to ROS's network implementation. Another preliminary test was run to determine what was causing the abnormal behavior. Baxter was commanded to follow the same 2D sinusoidal path as the previous test. However, data was recorded in three different ways. The first data recording was done using the Python ROS subscriber. The second data recording was done using the C++ ROS subscriber. The last recording method was using the rosbag utility to directly record data from the ROS topic. Figure 3.7 shows the position data recorded using the three different methods plotted against time. From Figure 3.7, it can be seen that the rosbag utility and the C++

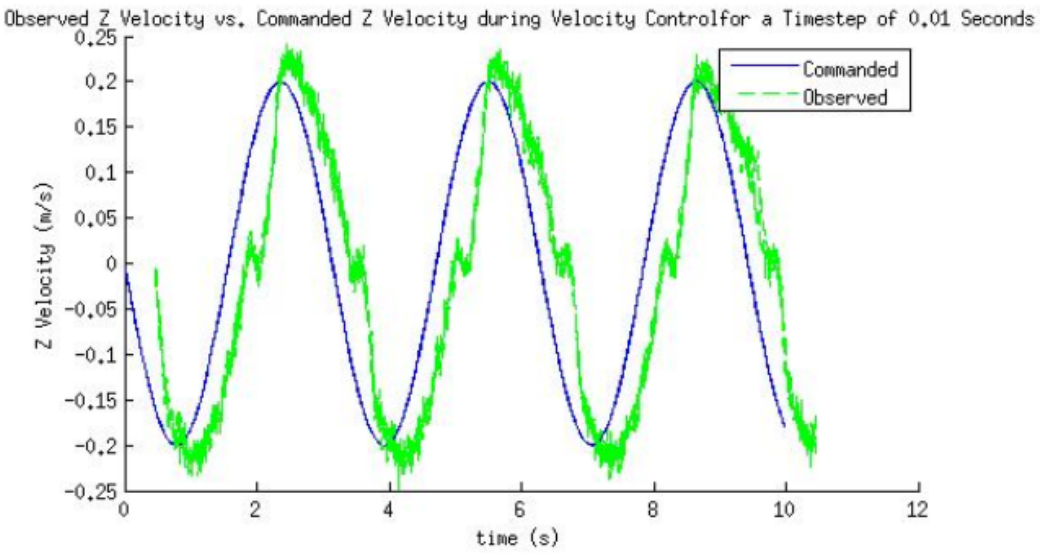

Figure 3.6: Commanded and Recorded End Effector Z Velocity Data

recording match exactly for all samples. The Python recording has missing samples and sometimes has bursts of data that arrive almost simultaneously. This is further demonstrated in Figure 3.8 where the time a data sample was sent by Baxter is plotted against the time the data sample was received by the test program. The time a data sample was sent is recorded from the header of the ROS messages which are generated by Baxter. The time a data sample was received is generated by the testing program on data reception. Again, certain Python samples are missing while some are received late and end up in a burst of data samples. The standard deviation of the difference between when data samples were sent and when the data was received revealed a much larger range of delays with Python than with C++ or rosbag. The standard deviation and mean of the difference between sending and receiving of data are shown in both Figures 3.7 and 3.8. The black line in Figure 3.8 shows the points where sending and receiving of data has no delay. The mean difference is not very useful in this case due to the presence of network delay.

Due to the issues found with Python in the preliminary testing, it was decided that a C++ library would be developed for interfacing with Baxter. This is possible due to the middleware nature of ROS where all ROS nodes functionality are abstracted away from other nodes [12]. That abstraction allows ROS nodes to be programmed using Python, C++, or LISP. The C++

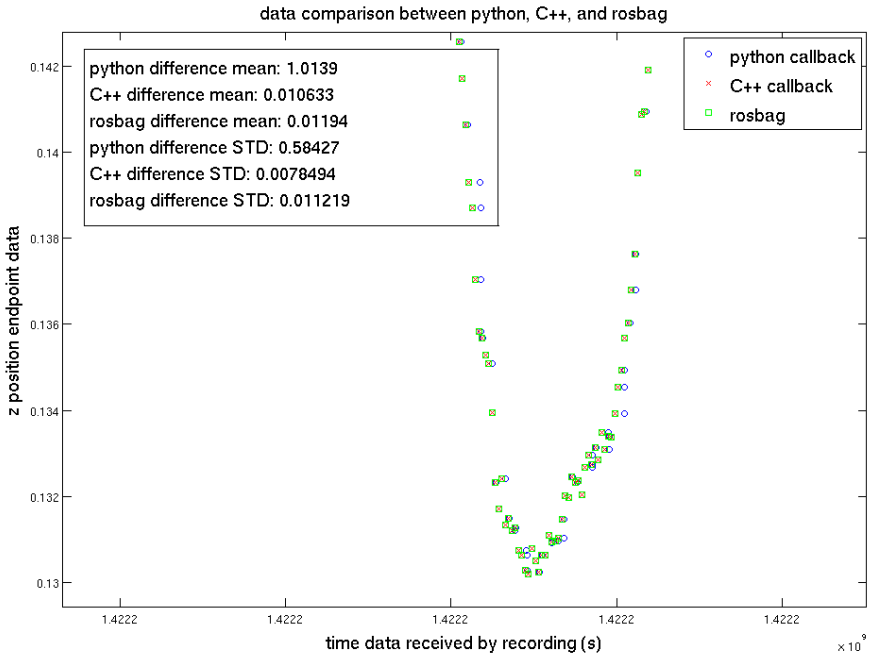

Figure 3.7: Data Recording Comparison Between Python, C++, and Rosbag

library mirrors the functionality of the Python SDK that Rethink Robotics provides and is still under development. The only difference from the Python SDK is the ability to pass user defined control systems to command Baxter. The user passes a function that has predefined input of Baxter's sensor data in the form of ROS messages and output of Baxter's joint commands for either position, velocity, or torque control. That function will execute at a user defined period to act as an external control system. Since Baxter's message publishing is effective up to 800 Hz, the effectiveness of the user control system will be limited by this. This feature was added because the data sampling issue with Python will most affect attempts at programming a control system to use with Baxter.

3.1.1.3 Repeatability Testing

Despite the issues seen in preliminary tests with Python and development of a C++ library, Python was chosen as the primary programming language for the repeatability testing. This is because the default performance of Baxter is being characterized so most of Baxter's default systems that the man-

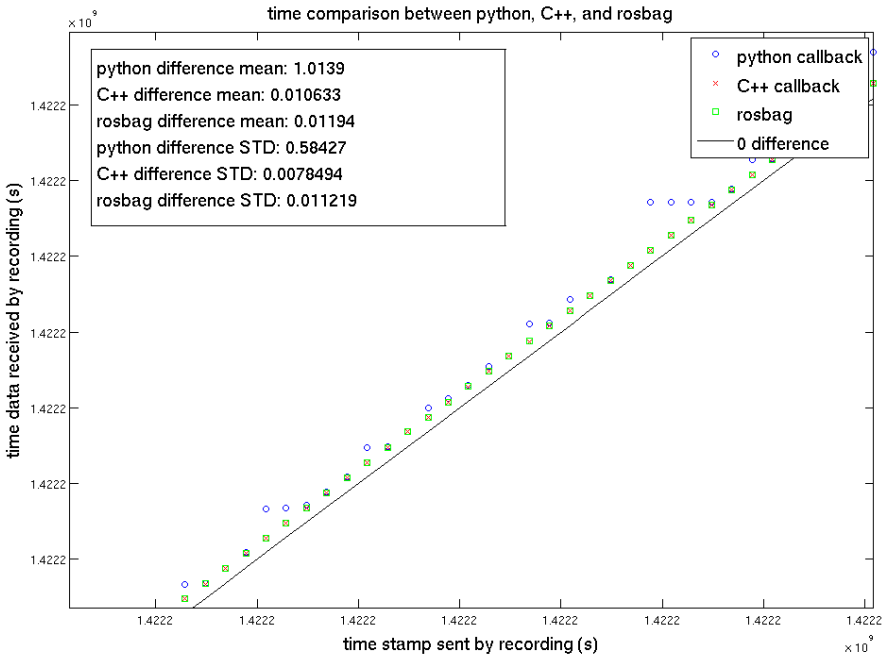

Figure 3.8: Time Data Sent and Received Comparison Between Python, C++, and Rosbag

ufacturer provides will be used including Rethink Robotic's Python SDK. Therefore, the Python SDK will be considered to offer the default performance of Baxter. Also, the repeatability testing does not rely on real time feedback from Baxter, so the Python issues are less of a concern.

For the repeatability testing, test poses are commanded using Baxter's position control mode. Data recording is done using the rosbag Python library to record to rosbag files. While the Python SDK provides subscribers to Baxter's data topics, there is no subscriber for motion capture data. So, a subscriber is created to listen for motion capture data so that all data recording can be handled by the main test program.

Since payload weight affects repeatability, payload weight will be varied during the experiment [15]. ISO 9283 specifies that the tested payloads should be 100% and 10% of the rated payload of the robot [19]. Baxter's rated maximum payload is 2.27 kg; therefore the tested weights will be 2.27 kg and 0.23 kg. 0.23 kg iron bars were used as the payload weights. For 2.27 kg payload testing, ten iron bars were duct taped around Baxter's wrist (see Figure 3.9). Three bars were taped on the top and bottom and two bars were taped on the sides of the wrist. This configuration is to keep

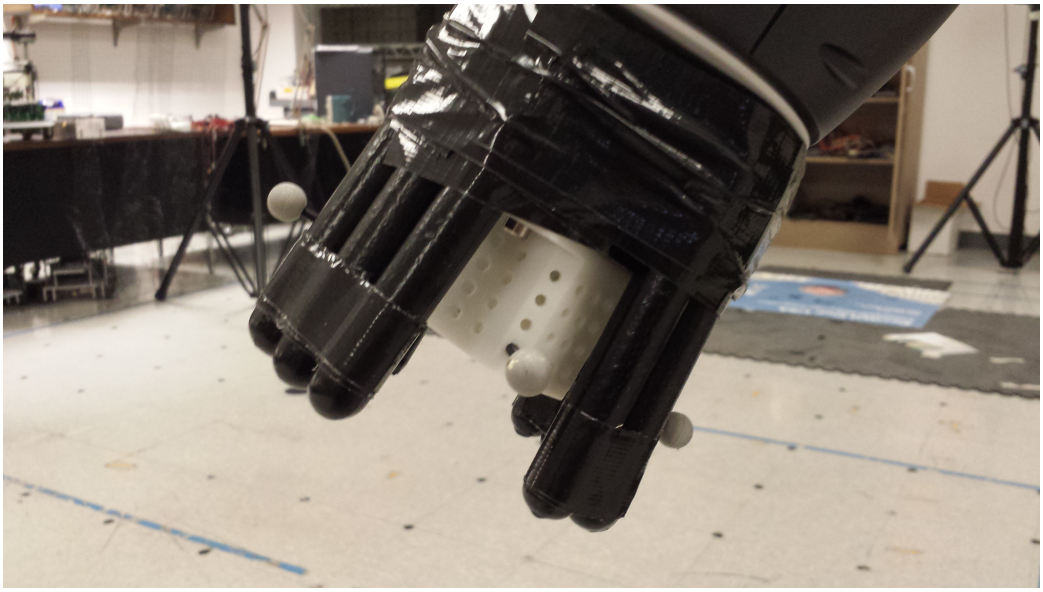

Figure 3.9: 100% Payload

the center of gravity of the payload still in the center of Baxter's wrist. The sides of Baxter's wrist (cuff) activate a zero-G mode which enables the arms to move freely using only gravity compensation torques [30]. Since this interferes with manipulation of Baxter's arms, cuff interaction was disabled to disable zero-G mode. This is done by publishing an empty message to the suppress_cuff_interaction ROS topic at 10 Hz. For the 0.23 kg payload, a single iron bar is attached centered on top of the fixture (see Figure 3.10).

ISO 9283 also specifies the variation of movement speed alongside payload. ISO 9283 requires the testing of 100%, 50%, and 10% of manufacturer rated maximum movement speed. Normally, Baxter limits its movement speed to 30% of its maximum movement speed. During testing, the speed was changed using the set_joint_position_speed function that is part of the limb class of the SDK Rethink Robotics developed where a percentage ratio is commanded as input. However, there is currently no manufacturer given specification for the maximum speed of Baxter's movement.

3.1.2 Motion Capture System

The motion capture system used is manufactured by OptiTrack. It consists of twenty-four Flex 3 motion capture cameras and four OptiHub 2 USB synchronization devices. The cameras surrounded an approximately 5.2 m

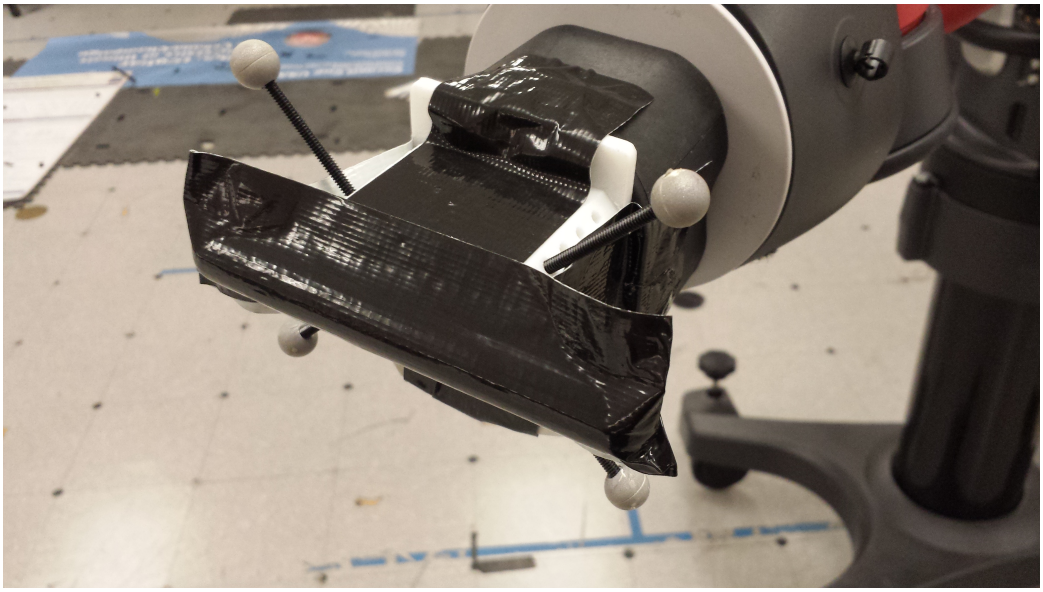

Figure 3.10: 10% Payload

by 7.2 m large area where testing took place. The system was connected to a Windows 8.1 desktop computer running OptiTrack's Motive software. The cameras are run at their maximum frame rate of 100 Hz. Since the OptiTrack system relies on IR light for motion capture, OptiTrack's reflective tape covered fiducial markers are used to provide the measurement point. Four of these markers form a rigid body whose position and orientation are measured by the system.

Instead of Baxter's default end effectors (mechanized grippers), a custom made fixture was attached to the end of Baxter's wrist (see Figure 3.11). This fixture was designed to hold the motion capture's fiducial markers to reduce the variance of the measurement due to shifting of the markers. The fixture was 3D printed and attached to Baxter's wrist using screws at Baxter's end effector mounting points. Fiducial markers were attached to the fixture using threaded rods.

The reconstructed 3D pose data of the rigid body assigned to the four fiducial markers is streamed over the LAN network using OptiTrack's NatNet streaming protocol. A ROS node on the Linux desktop intercepts the streamed motion capture data and rebroadcasts it over the ROS network as a ROS topic. This ROS node is a modified version of Clearpath Robotic's ROS NatNet code [31]. Clearpath's NatNet node needed to be modified due to being designed to process the outdated NatNet v2.2 protocol. The cur-

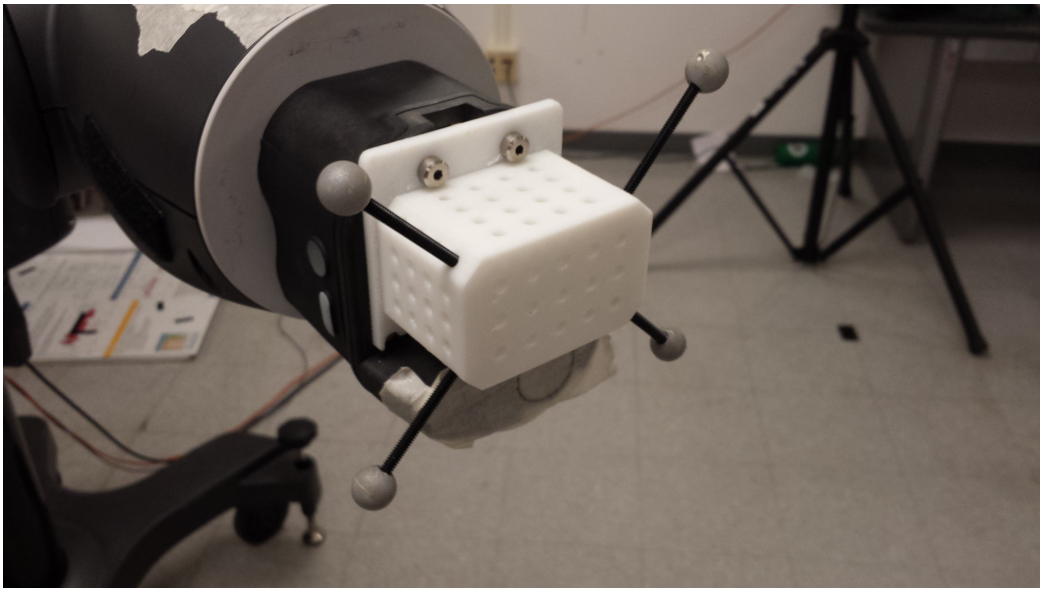

Figure 3.11: 3D Printed End Effector

rent OptiTrack system was running NatNet v2.7 which had rearranged the order of data packets streamed from v2.2. Clearpath's NatNet ROS node was modified to parse the data packets as per NatNet v2.7's specifications.

OptiTrack's motion capture system does not have any timing information about the exact moment when the cameras recorded data. It only provides information on when data was broadcasted over the NatNet protocol. However, the OptiHub 2 has a synchronization out Bayonet Neill–Concelman (BNC) port. This synchronization out port pulses a digital 5V signal every time the cameras' sensors are exposed. Therefore, timing information could be obtained about the exposure time of the system if timestamps were assigned to the synchronization out pulses.

A program was developed on a Raspberry Pi mini computer to assign timestamps to these synchronization pulses. The Raspberry Pi was set up to run a barebones operating system distribution called Machinoid and utilized the Xenomai real-time Linux framework. The Xenomai framework is a secondary kernel that runs alongside the default Linux kernel. The Xenomai kernel handles all real-time processes and can preempt the Linux kernel if needed to ensure the real-time nature of certain tasks. A two-part program was developed to assign timestamps to the OptiHub synchronization pulses. A program in the Xenomai kernel would fire an interrupt whenever the 5 V logic pulse from the OptiHub was detected on a general purpose input output

(GPIO) pin. Then a timestamp from the Raspberry Pi's system clock would be recorded. The Raspberry Pi's system clock would be synchronized with a local time system via network time protocol (NTP) or precision time protocol (PTP). A separate program outside of the kernel in user space would query the kernel application for the timestamp and record it to a file. The recorded timestamps could then be recombined with the motion capture data after any experiments were done running.

The recombination of timestamps and the motion capture data was dependent on the first timestamp and motion capture sample being assigned to each other. Each successive timestamp and data sample could then be associated. However, the motion capture system could drop a frame for a variety of reasons. A dropped frame would result in the synchronization pulse still firing, which meant a timestamp would exist but the data sample would not have been recorded. In order to account for dropped frames, the period of each timestamp and motion capture data sample was examined. Since the cameras fire at 100 Hz, any period between motion capture data larger than 0.01 s meant a timestamp would be discarded.

For the repeatability experiments, this timestamping system was not utilized. This was mainly due to the length of testing. The recombination of time data with motion capture data required all data samples to be recorded. Unfortunately, recording all motion capture data during the entirety of repeatability testing was infeasible due to the amount of time testing required. Therefore, motion capture data was only recorded when Baxter's arm was done executing its movement command and had settled, giving the motion capture system time to process the data and broadcast it over LAN. This made recombination of timestamps and motion capture data impossible. However, it is recommended to utilize this timestamping system whenever possible to obtain the most accurate time of measurement for the motion capture system. A future improvement to the system would also be to intercept motion capture data, assign the appropriate timestamp, and then rebroadcast it again.

Table 3.1: Test Pose Coordinates (m)

Point	Left	Right
P1	(0.68, 0.68, 0.25)	(0.68, -0.68, 0.25)
P2	(0.80, 0.80, 0.37)	(0.80, -0.80, 0.37)
P3	(0.80, 0.56, 0.37)	(0.80, -0.56, 0.37)
P4	(0.56, 0.56, 0.13)	(0.56, -0.56, 0.13)
P5	(0.56, 0.80, 0.13)	(0.56, -0.80, 0.13)

3.2 Measurement Procedure

The measurement procedure is based mainly on the method presented in the ISO 9283 standard. The ISO 9283 repeatability evaluation varies target location, velocity, and payload, all of which affect repeatability [19, 15, 16].

As in accordance with the ISO 9283 standard, a test cube is defined. The test cube chosen has 30 cm sides. The corner of the test cube closest to Baxter's origin (C_7 from Figure 2.1) has the coordinates of (0.53, 0.53, 0.1) for the left arm and (0.53, -0.53, 0.1) for the right arm. The actual selected test point coordinates (see Figure 2.2) for both arms are shown in Table 3.1. A visualization of the test poses is shown in Figure 3.12. Since the orientation of poses is not specified by ISO 9283, an orientation of the normal of the end effector being perpendicular to the test plane was chosen. All poses have the orientation shown in quaternion Equation (3.1). Figure 3.13 shows the test poses in Baxter's workspace for the left arm.

$$H = 0.3827 + 0i + 0.9239j + 0k \tag{3.1}$$

Before executing the measurement procedure, both Baxter and the motion capture system need to be calibrated. Baxter has a calibration and tare routine for its arms to calibrate torque sensors, gravity compensation, and the effects of the large spring on the S1 joint [33]. The motion capture system calibration consists of using a wand with fiducial markers attached to it at known lengths. By moving the wand through the motion capture space, relative camera positions are obtained and allow for the reconstruction of fiducial markers' 3D positions. A ground plane is also used to set the motion capture's world reference coordinate frame.

For the measurement procedure, Baxter's arms are commanded to the various test points. Starting with the left arm, Baxter's wrist cycles from P_1

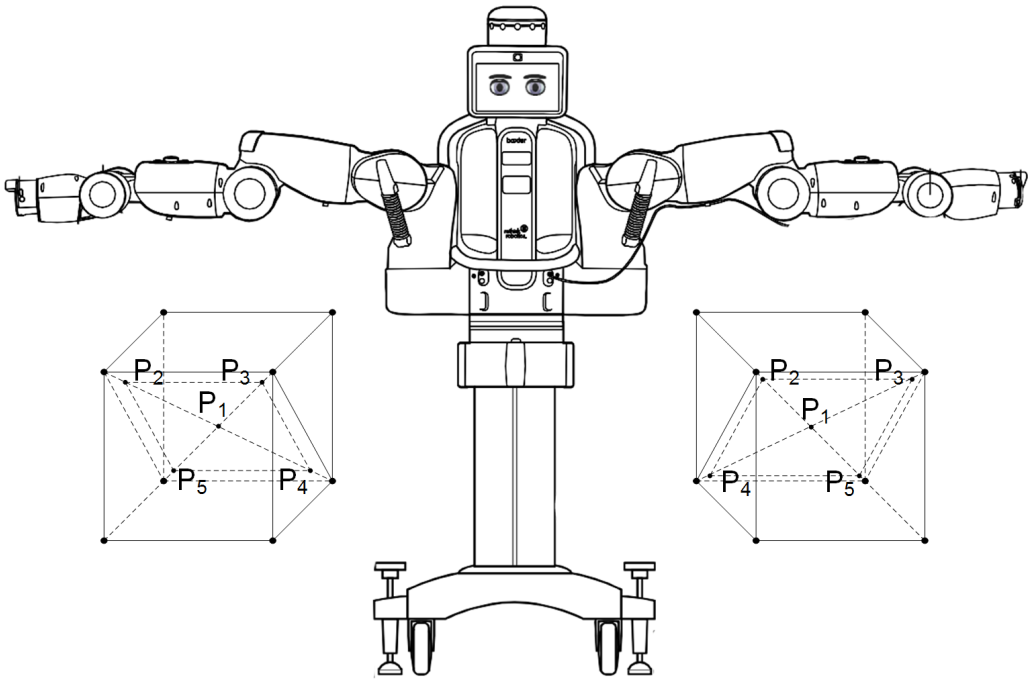

Figure 3.12: Test Poses for Baxter's Left and Right Arms; Not to Scale. Adapted from Baxter Wiki [32]

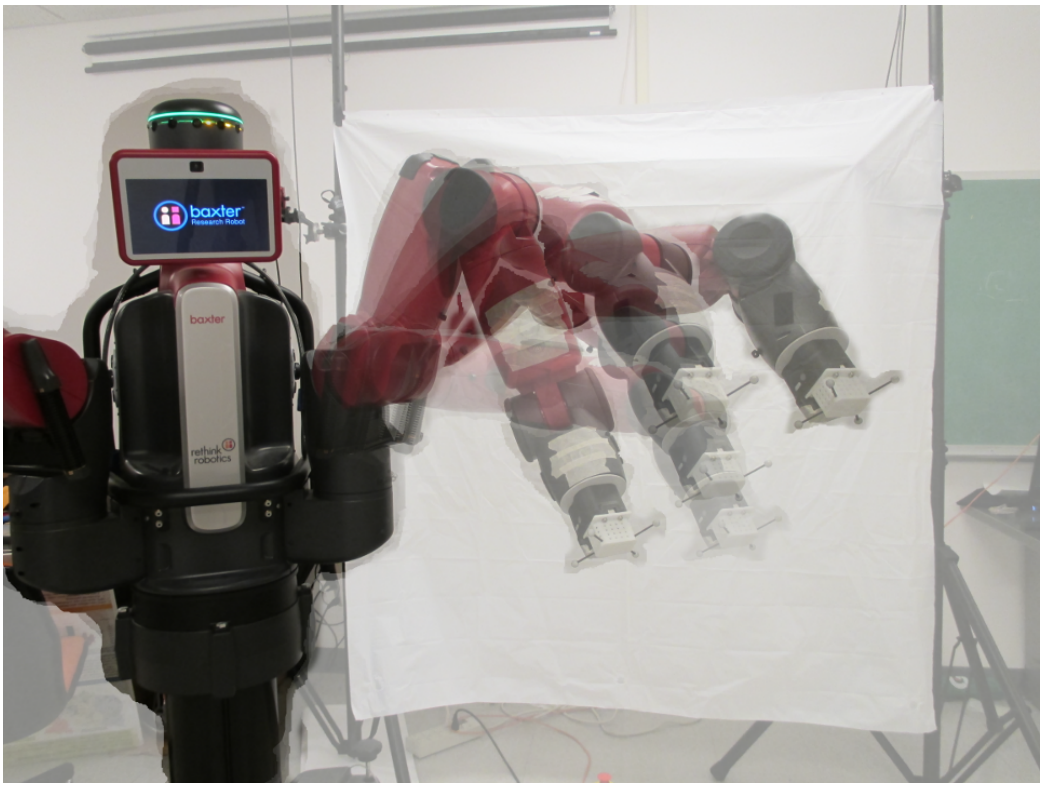

Figure 3.13: Overlay of Test Poses in Baxter's Workspace

through P_5. This is done for 30 complete cycles for each set of payload and speed parameters. At the beginning of each trial, Baxter calculates the joint angles necessary to achieve the target poses through its inverse kinematics service.

For each trial of 30 cycles, payload and velocity are varied. First, trials with 2.27 kg payload attached to Baxter's end effector are executed with 100%, 50%, and 10% of Baxter's maximum position control velocity. Then trials with 0.23 kg payload are done with the same velocities. The whole process is executed for the left arm and then the right arm, attaching the same end effector fixture and payload weights.

The overall testing routine is shown in Algorithm 2. Algorithm 3 shows the subroutine of repeatability measurement for each set of parameters. The other subroutines are provided by Rethink Robotic's Python SDK. The subroutine set_joint_position_speed sets Baxter's movement speed to a percentage of its max movement speed. The inverse_kinematics subroutine is the inverse kinematics service of Baxter which returns the joint angles necessary to reach a passed input pose. The write function of rosbag is the function for writing to rosbag files and is part of the rosbag library. Attaching the payload to Baxter's arms also includes attaching the custom fixture that holds the fiducial markers. The rigid body is redefined for each set of testing in Motive due to the drifting error of the motion capture. By resetting the rigid body, the error of the motion capture system resets as well. The inverse kinematics are only invoked for each round of repeatability testing once. This is because Baxter's inverse kinematics are seeded by the current physical configuration of its arms and therefore may produce a different solution if the inverse kinematics were solved after every movement. So joint angle solutions for each test pose are stored for that round of testing and used for each of the 30 movements to the five test poses.

3.3 Data Analysis

The recorded motion capture data is analyzed using the same method as presented in ISO 9283 as seen in Section 2.3. Repeatability is calculated separately as position repeatability and orientation repeatability.

For position repeatability, the barycentre of each test point for each set of

Algorithm 2: Overall Repeatability Testing Routine

Attach 100% payload to Baxter's left arm;
Reset motion capture rigid body;
LeftArm.set_joint_position_speed(100%);
Repeatability_subroutine(Left, P_1 **to** P_5);
LeftArm.set_joint_position_speed(50%);
Repeatability_subroutine(Left, P_1 **to** P_5);
LeftArm.set_joint_position_speed(10%);
Repeatability_subroutine(Left, P_1 **to** P_5);

Attach 10% payload to Baxter's left arm;
Reset motion capture rigid body;
LeftArm.set_joint_position_speed(100%);
Repeatability_subroutine(Left, P_1 **to** P_5);
LeftArm.set_joint_position_speed(50%);
Repeatability_subroutine(Left, P_1 **to** P_5);
LeftArm.set_joint_position_speed(10%);
Repeatability_subroutine(Left, P_1 **to** P_5);

Attach 100% payload to Baxter's right arm;
Reset motion capture rigid body;
RightArm.set_joint_position_speed(100%);
Repeatability_subroutine(Right, P_1 **to** P_5);
RightArm.set_joint_position_speed(50%);
Repeatability_subroutine(Right, P_1 **to** P_5);
RightArm.set_joint_position_speed(10%);
Repeatability_subroutine(Right, P_1 **to** P_5);

Attach 10% payload to Baxter's right arm;
Reset motion capture rigid body;
RightArm.set_joint_position_speed(100%);
Repeatability_subroutine(Right, P_1 **to** P_5);
RightArm.set_joint_position_speed(50%);
Repeatability_subroutine(Right, P_1 **to** P_5);
RightArm.set_joint_position_speed(10%);
Repeatability_subroutine(Right, P_1 **to** P_5);

Algorithm 3: Subroutine for Individual Repeatability Measurement

input : Baxter_limb, P_1 **to** P_5
output: Rosbag file with measurements and joint angles

```
Move_to_neutral(Baxter_limb);
P1_joints ← inverse_kinematics(P1);
P2_joints ← inverse_kinematics(P2);
P3_joints ← inverse_kinematics(P3);
P4_joints ← inverse_kinematics(P4);
P5_joints ← inverse_kinematics(P5);
for c ← 1 to 30 do
    Move_to_position(Baxter_limb, P1_joints);
    Sleep 2s;
    Rosbag.write(joint_angles, measurement);
    Move_to_position(Baxter_limb, P2_joints);
    Sleep 2s;
    Rosbag.write(joint_angles, measurement);
    Move_to_position(Baxter_limb, P3_joints);
    Sleep 2s;
    Rosbag.write(joint_angles, measurement);
    Move_to_position(Baxter_limb, P4_joints);
    Sleep 2s;
    Rosbag.write(joint_angles, measurement);
    Move_to_position(Baxter_limb, P5_joints);
    Sleep 2s;
    Rosbag.write(joint_angles, measurement);
```

parameters is calculated using Equation (2.5). With the barycentre coordinates, the deviation of each test location from the barycentre coordinates is calculated with Equation (2.6). With those deviations and Equations (2.7) and (2.8), a sphere can be obtained. The radius of this sphere (RP_l) is the position repeatability for each set of target location, speed, and payload parameters and is obtained using Equation (2.8).

Orientation repeatability is calculated similarly to position repeatability but is based purely on the standard deviation of the orientation angles. Since orientation can be represented by three rotation angles, orientation repeatability is divided into the repeatability of each rotation angle (RP_z, RP_y, RP_x). The rotation angles are defined in terms of the motion capture's base frame rather than Baxter's base frame. The motion capture's base frame is shown in Figure 3.14 in relation to Baxter's base frame. The motion capture's x axis is anti-parallel to Baxter's x axis. The motion capture's y and z axes are parallel to Baxter's z and y axes respectively. Figure 3.14 does not show the translation between the two base frames. Therefore RP_z represents rotation about Baxter's y axis, RP_y represents rotation about Baxter's z axis, and RP_x represents negative rotation about Baxter's x axis.

Motion capture data orientation is represented using quaternions, so it is necessary to first convert the quaternions to rotation angles. The quaternions are converted to the rotation angles of the rotation sequence ZYX. Also, the motion capture represents rotations in the range of -π to π. Since calculations of orientation repeatability rely on a summation, the sign change of the rotation range will cause issues. So, the range of rotations is first converted to a range of 0 to 2π. Using Equation (2.9), repeatability for each individual rotation angle of the orientation is calculated.

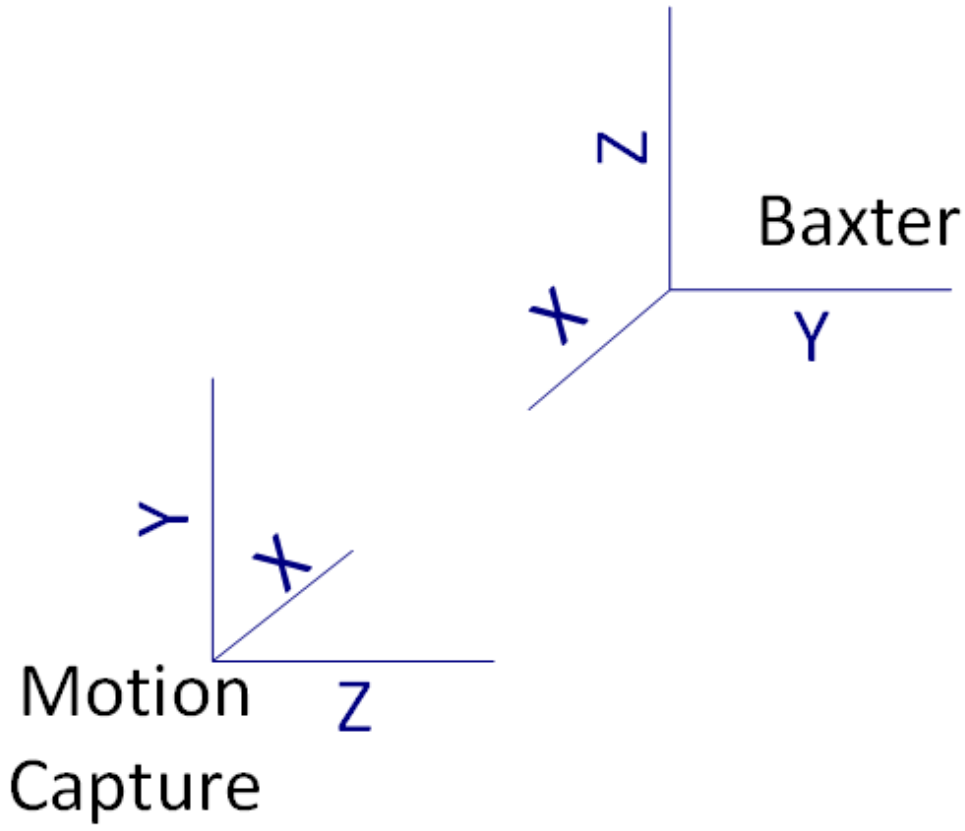

Figure 3.14: Relationship Between Axes of the Motion Capture's and Baxter's Base Frames; Translation not to Scale

CHAPTER 4

RESULTS

This chapter will look at the repeatability results obtained from tests done with Baxter. These results are split into position and orientation repeatability.

4.1 Position Repeatability

Tables 4.1 and 4.2 show the calculated position repeatability for the left and right arms, respectively, under the different test conditions.

From Tables 4.1 and 4.2, the lowest position repeatability for the left and right arms, respectively, is 1.1 mm and 1.3 mm. For the left arm, RP_l of 1.1 mm occurs at target location P_4, with a 10% load, and at 50% speed. For the right arm, RP_l of 1.3 mm occurs at target location P_4, with 10% load, and at 10% speed. Figures 4.1 and 4.2 show the barycentre spheres with measurement points of these lowest repeatability for the left and right arms respectively. The deviations of each data sample for the left and right arms are shown in Figures 4.3 and 4.4 respectively.

The minimum repeatability is the best case scenario which can be measured under only certain conditions. Depending on the speed, payload weight, or target locations, the repeatability changes. Therefore it is more meaningful to look at the mean repeatability rather than the minimum repeatability for determining an overall specification. However, if it is necessary for Baxter to operate under predetermined parameters, it is better to look at the repeatability for that specific set of parameters. The average repeatability was determined by calculating the repeatability for the entire data set across all parameters with deviations calculated from the respective barycentre of each parameter set. For the left arm, the average repeatability is 2.9 mm with a standard deviation of 0.7 mm. The right arm has an average repeatability

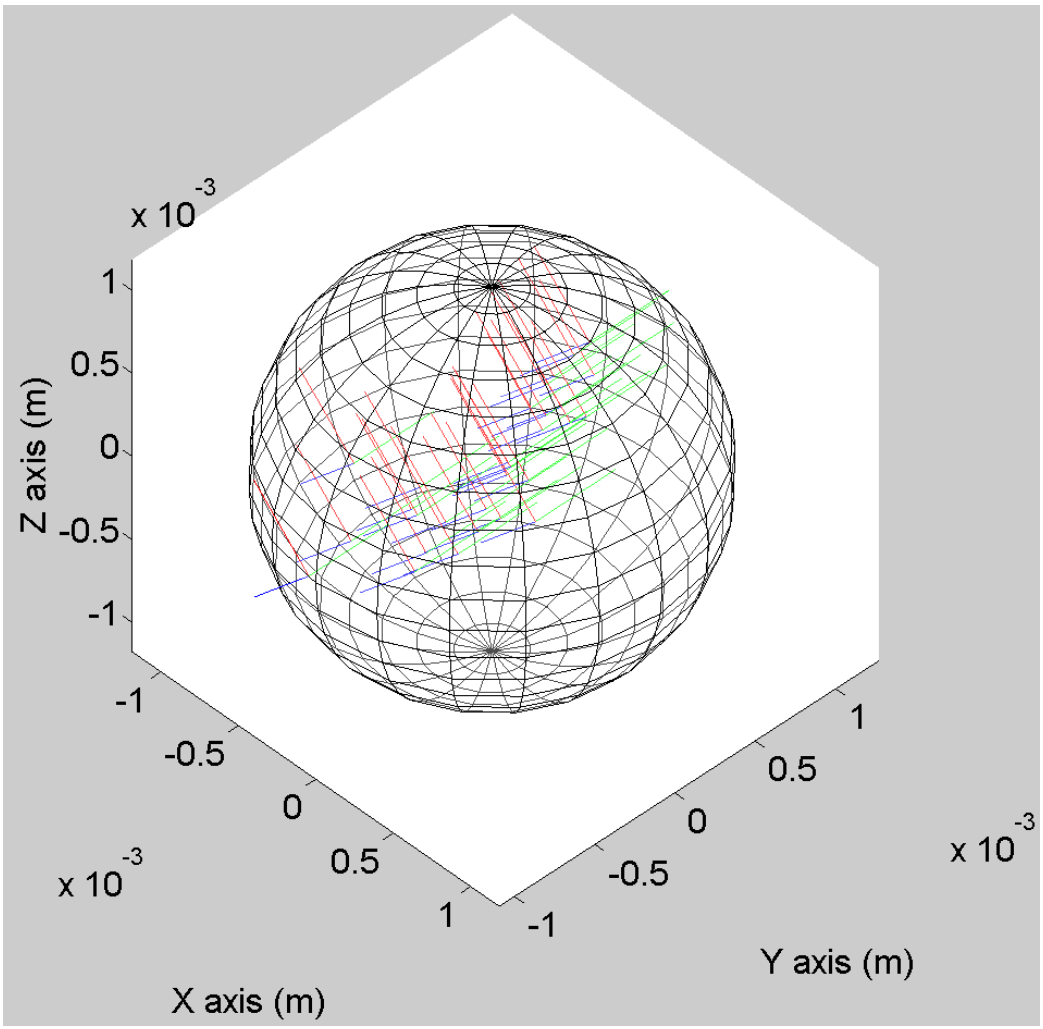

Figure 4.1: Left Arm Barycentre Sphere with Measurement Points of Best Repeatability

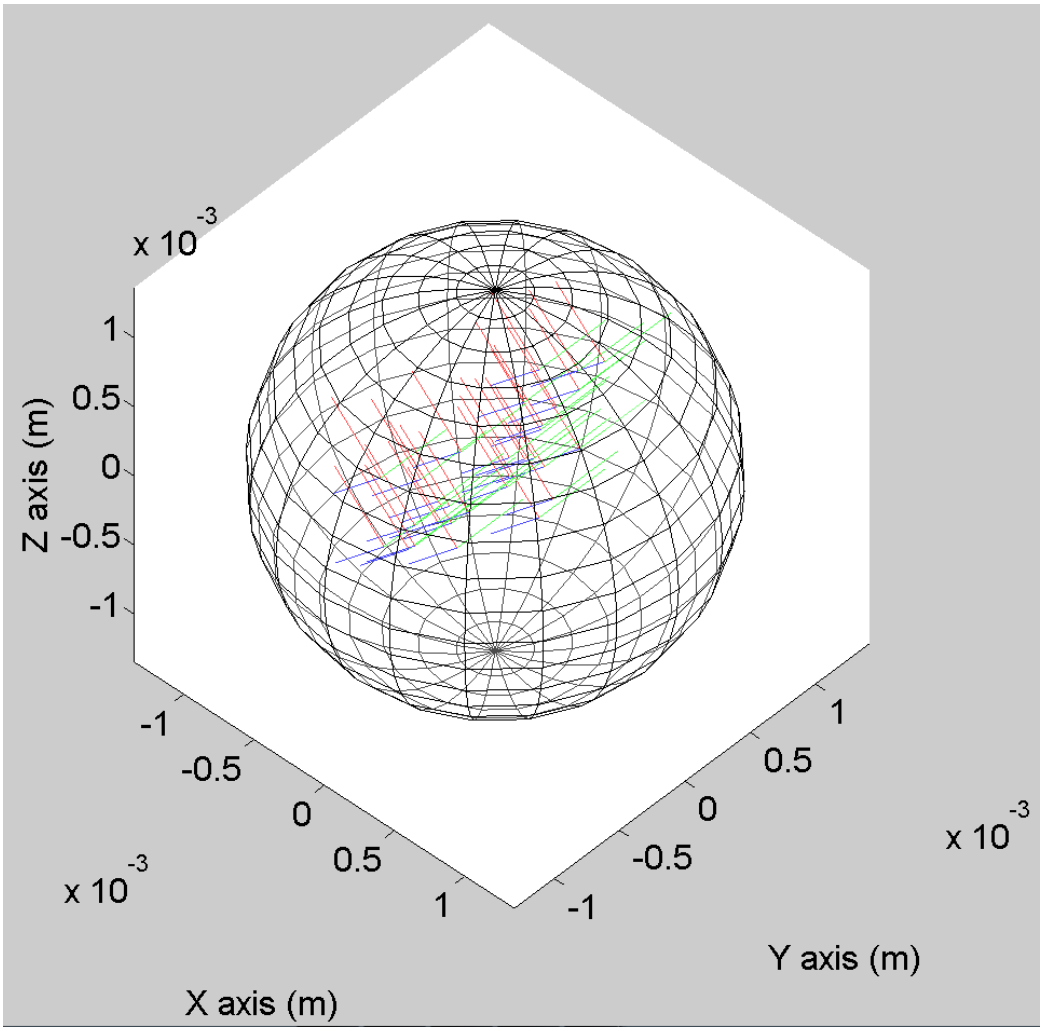

Figure 4.2: Right Arm Barycentre Sphere with Measurement Points of Best Repeatability

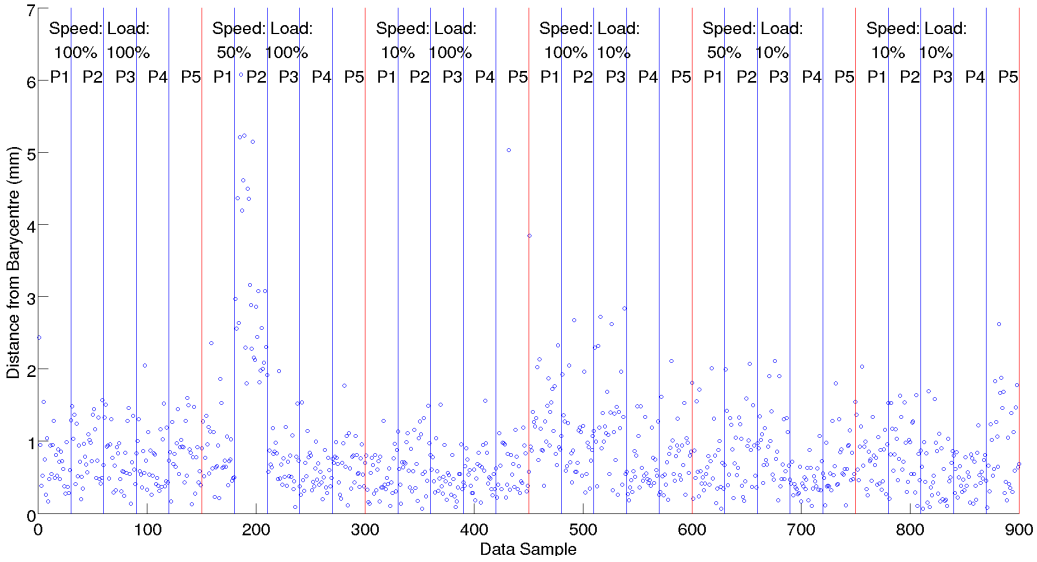

Figure 4.3: Left Arm Deviation of Measurements from Barycentre. Blue Lines Separate Poses. Red Lines Separate Parameter Sets

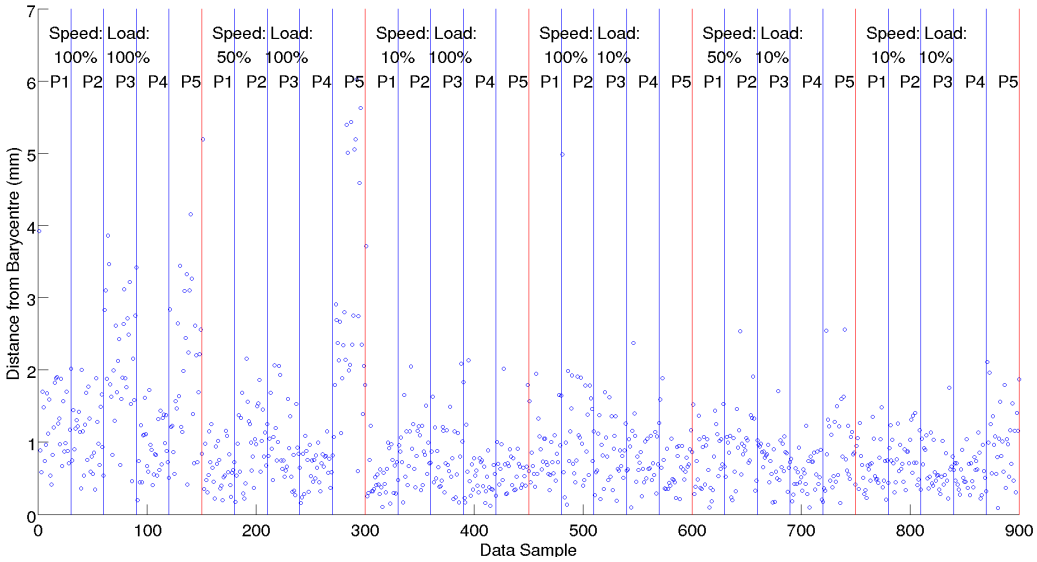

Figure 4.4: Right Arm Deviation of Measurements from Barycentre. Blue Lines Separate Poses. Red Lines Separate Parameter Sets

Table 4.1: Position Repeatability (mm) of Left Arm under Varied Parameters

Speed	Load	P1	P2	P3	P4	P5
100%	100%	2.1	2.0	1.9	2.0	2.0
50%	100%	2.3	6.8	1.9	1.5	1.7
10%	100%	1.5	1.8	3.4	1.8	2.2
100%	10%	3.3	2.7	3.4	1.8	2.2
50%	10%	2.0	2.4	2.4	1.1	1.9
10%	10%	2.0	2.2	1.9	1.3	2.9

Table 4.2: Position Repeatability (mm) of Right Arm under Varied Parameters

Speed	Load	P1	P2	P3	P4	P5
100%	100%	3.4	2.5	4.7	2.1	4.9
50%	100%	3.4	2.5	2.5	1.4	7.6
10%	100%	2.5	2.1	2.3	1.7	1.9
100%	10%	2.1	3.9	2.0	2.1	1.8
50%	10%	2.0	2.4	1.7	1.4	2.7
10%	10%	1.7	1.8	1.7	1.3	2.6

of 3.3 mm with a standard deviation of 0.8 mm.

For the left arm, the mean repeatability for 100%, 50%, and 10% speed is 2.2 mm, 2.4 mm, and 2.0 mm respectively. For the right arm, the mean repeatability for 100%, 50%, and 10% speed is 3.0 mm, 2.8 mm, and 2.0 mm respectively. From Offodile's results, he showed that higher speeds, especially beyond 50% speed, degrade repeatability [15]. Both the left and right arm show the best repeatability at 10% speed while the mean repeatability became worse for 50% and 100% speed. Although, the left arm demonstrated a better mean repeatability at 100% than at 50% speed.

Offodile also showed that higher payload weights resulted in worse repeatability [15]. Baxter showed a mean repeatability of 2.2 mm at both 100% load and 10% load in its left arm and of 3.0 mm and 2.0 mm in its right arm. The right arm results agree with Offodile. The left arm shows the same repeatability for both payloads; however, there is a higher standard deviation for the 100% load of 1.4 mm while the 10% load only has a standard deviation of 0.7 mm.

Riemer's results suggested that the farther away a target location was from the robot, the worse the repeatability measured [16]. Under ISO 9283,

Table 4.3: Orientation Repeatability (rad) of Left Arm under Varied Parameters

Angle	Max	Mean	Min	StD
RP_z	0.0149	0.0047	0.0029	0.0021
RP_y	0.0075	0.0030	0.0014	0.0013
RP_x	0.0207	0.0043	0.0027	0.0032

this would mean that P_4 should give the best repeatability while P_2 would give the worst repeatability. On average, Baxter's left arm demonstrated the best repeatability at P_4 with 1.5 mm repeatability. The worst average repeatability for the left arm was at P_2 with 3.0 mm. For Baxter's right arm, the best was at P_4 with 1.7 mm but the worst was at P_5 with 3.6 mm. Generally, the results of testing seemed to agree with Riemer's findings. However, the right arm achieved its worst repeatability at P_5 which does not follow Riemer's findings. P_5 also happened to yield a much higher standard deviation of 2.3 mm, which suggests some irregular data samples made the mean repeatability worse.

Overall, Baxter's positional repeatability was determined to be 2.9 mm to 3.3 mm on average. However, depending on speed, payload, and target location, the repeatability could range from 1.1 mm to 7.6 mm. It is also important to note that the positional accuracy of the OptiTrack motion capture system was 0.2 mm when testing. Although that accuracy varies during testing and also drifts to a worse accuracy as time progresses, the accuracy was always sub-millimeter.

4.2 Orientation Repeatability

Orientation repeatability is divided among the three rotation angles (Z,Y,X). These rotation angles are defined in relation to the motion capture's base coordinate frame rather than Baxter's coordinate frame.

Tables 4.3 and 4.4 show the range of repeatability for each rotation angle for the left and right arms. For the left arm, the average angular repeatability ranges from 0.003 to 0.0047 radians while the average repeatability for the right arm ranges from 0.0037 to 0.0054 radians.

Unfortunately, Offodile and Riemer ignore orientation in their work, mostly

Table 4.4: Orientation Repeatability (rad) of Right Arm under Varied Parameters

Angle	Max	Mean	Min	StD
RP_z	0.0077	0.0054	0.0027	0.0014
RP_y	0.0073	0.0037	0.0018	0.0012
RP_x	0.0152	0.0049	0.0018	0.0025

Table 4.5: Average Angular Repeatability (rad) at Various Speeds

Arm	Angle	100%	50%	10%
Left	RP_z	0.0051	0.0053	0.0038
Left	RP_y	0.0037	0.0027	0.0025
Left	RP_x	0.0046	0.0051	0.0033
Right	RP_z	0.0062	0.0054	0.0045
Right	RP_y	0.0041	0.0039	0.0032
Right	RP_x	0.0055	0.0056	0.0035

due to limitations of their measurement devices [15, 16]. Under ISO 9283, orientation repeatability is calculated so Baxter's orientation repeatability was compared against the various conditions of testing [19]. Table 4.5 shows the average angular repeatability at different speeds. As with positional repeatability, it seems Baxter performs better at lower speeds, consistently achieving better repeatability at 10% speed. Table 4.6 shows average repeatability at different payload weights. Across all angles on both arms, repeatability is better with the lighter load. Table 4.7 shows the repeatability at the five target locations. The best orientation repeatability occurs at P_4, which is the closest point to Baxter. Most of the worst repeatability occurs at P_2, the farthest point, but some also occurs at other target locations with the right arm. From this, it would seem that orientation repeatability follows the same trends as positional repeatability, but a separate study would need to be done to conclude this.

It is important to note that there is no measure of the OptiTrack motion capture system's rigid body orientation accuracy. Therefore, it is hard to conclude with an absolute orientation repeatability measurement. However, it would tentatively seem that the orientation accuracy is in the range of 0.003 to 0.005 radians.

Table 4.6: Average Angular Repeatability (rad) at Various Loads

Arm	Angle	100%	10%
Left	RP_z	0.0049	0.0046
Left	RP_y	0.0027	0.0016
Left	RP_x	0.0049	0.0038
Right	RP_z	0.0055	0.0053
Right	RP_y	0.0043	0.0032
Right	RP_x	0.0055	0.0042

Table 4.7: Average Angular Repeatability (rad) at Various Target Locations

Arm	Angle	P1	P2	P3	P4	P5
Left	RP_z	0.0047	0.0062	0.0042	0.0041	0.0045
Left	RP_y	0.0033	0.0032	0.0031	0.0024	0.0029
Left	RP_x	0.0042	0.0068	0.0038	0.0033	0.0036
Right	RP_z	0.0062	0.0050	0.0051	0.0045	0.0062
Right	RP_y	0.0040	0.0034	0.0044	0.0029	0.0040
Right	RP_x	0.0052	0.0048	0.0044	0.0035	0.0064

CHAPTER 5

CONCLUSION

The Baxter robot from Rethink Robotics has a rated accuracy of ±5 mm but lacks a repeatability measurement. This thesis attempts to run characterization tests to measure the repeatability of Baxter using a motion capture device.

Past literature has found that many factors influence repeatability including speed of movement, payload weight, and target location [15, 16]. The test procedure in this thesis was based primarily on the methods set in ISO 9283 [19]. ISO 9283 was chosen as the primary test method because it varied the appropriate parameters that influence repeatability measurements. The repeatability was determined to be at best 1.1 mm and 1.3 mm for the left and right arms respectively. On average, the repeatability was determined to be 2.9 mm and 3.3 mm for the left and right arms respectively. Orientation repeatability was determined separately and found to be on average 0.003 to 0.0047 radians for the left arm and 0.0037 to 0.0054 radians for the right arm. However, since no orientational accuracy specification is known about the motion capture system, this result is inconclusive.

Under the method of ISO 9283, Baxter now has a repeatability measurement for both position and orientation. However, there remain many possible future studies of Baxter's abilities. Since it is shown that Baxter's repeatability changes under the various parameters of testing, Baxter's accuracy should also be retested under the various conditions following ISO 9283 standard. Also, Rethink Robotics only provides a rated positional accuracy. Therefore, an orientation accuracy should also be determined. Additionally, ISO 9283 specifies many measurements of robot performance such as path accuracy and repeatability, overshoot, or stabilization time [19]. These performance metrics could provide much more information on Baxter's abilities.

The measurement procedure of repeatability can also be improved. Since Baxter's repeatability is found to be in the millimeter range and the motion

capture accuracy was submillimeter, it would be advisable to use a measurement system with a higher measurement accuracy and a known orientation accuracy. The warm-up period was not considered in this work due to the large amount of time beforehand to calibrate Baxter and the motion capture system. However, the warm-up time should be determined to ensure no influence of warm-up. According to Jeswiet and Helferty, warm-up time should be determined via ANSI standards [28]. Another improvement is the number of cycles used to determine repeatability. Jeswiet and Helferty found that at least 200 cycles of testing gave better estimates of repeatability [28].

In a separate field of study, Baxter's accuracy and repeatability could be improved through control system design or kinematic calibration [8]. It would also be worthwhile to investigate the dynamic effects of Baxter's SEAs on its performance.

REFERENCES

[1] C. Fitzgerald, "Developing baxter," in *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, April 2013, pp. 1–6.

[2] J. Beggs, *Kinematics*. Hemisphere Publishing Corporation, 1983.

[3] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Hoboken (N.J.): John Wiley & Sons, 2006.

[4] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, ser. Advanced Textbooks in Control and Signal Processing. Springer, 2009.

[5] Wikipedia, the free encyclopedia, "File:classic-dhparameters.png," 2014, [Online; accessed July 1, 2015]. [Online]. Available: https://commons.wikimedia.org/wiki/File:Classic-DHparameters.png

[6] J. M. Hollerbach, "A survey of kinematic calibration," ser. The Robotics Review 1, O. Khatib, J. J. Craig, and T. Lozano-Pérez, Eds. Cambridge, MA, USA: MIT Press, 1989, pp. 207–242.

[7] S. Hayati and M. Mirmirani, "Improving the absolute positioning accuracy of robot manipulators," *Journal of Robotic Systems*, vol. 2, no. 4, pp. 397–413, 1985.

[8] B. Mooring, M. Driels, and Z. Roth, *Fundamentals of Manipulator Calibration*. New York, NY, USA: John Wiley & Sons, Inc., 1991.

[9] M. M. Williamson, "Series elastic actuators," M.S. thesis, Massachusetts Institute of Technology, 1995.

[10] "Hardware Specifications," 2014. [Online]. Available: http://sdk.rethinkrobotics.com/wiki/Hardware_Specifications

[11] "Baxter Research Robot Software Developers Kit (SDK)," 2014. [Online]. Available: http://sdk.rethinkrobotics.com/wiki/Baxter_Research_Software_Developers_Kit_%28SDK%29

[12] "ROS Introduction," 2014. [Online]. Available: http://wiki.ros.org/ROS/Introduction

[13] "ROS Concepts," 2014. [Online]. Available: http://wiki.ros.org/ROS/Concepts

[14] "API Reference," 2014. [Online]. Available: http://sdk.rethinkrobotics.com/wiki/API_Reference

[15] O. F. Offodile and K. Ugwu, "Evaluating the effect of speed and payload on robot repeatability," *Robotics and Computer-Integrated Manufacturing*, vol. 8, no. 1, pp. 27 – 33, 1991.

[16] R. Riemer and Y. Edan, "Evaluation of influence of target location on robot repeatability," *Robotica*, vol. 18, pp. 443–449, 7 2000.

[17] Y. Edan, L. Friedman, A. Mehrez, and L. Slutski, "A three-dimensional statistical framework for performance measurement of robotic systems," *Robotics and Computer-Integrated Manufacturing*, vol. 14, no. 4, pp. 307 – 315, 1998.

[18] W. Veitschegger and C.-H. Wu, "Robot accuracy analysis based on kinematics," *Robotics and Automation, IEEE Journal of*, vol. 2, no. 3, pp. 171–179, Sep 1986.

[19] International Organization for Standardization, *Performance Criteria and Related Test Methods: ISO 9283*, ser. ISO:International Standard, 1998.

[20] S. Nof, *Handbook of Industrial Robotics*, ser. Electrical and Electronic Engineering. Wiley, 1999.

[21] L. W. Shing and N. K. Loon, "Experimental and statistical determination of the static position repeatability and a recommended specification for a SCARA robot," *Robotics and Computer-Integrated Manufacturing*, vol. 9, no. 3, pp. 247 – 253, 1992.

[22] B. Preising and T. Hsia, "Robot performance measurement and calibration using a 3d computer vision system," in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, Apr 1991, pp. 2079–2084 vol.3.

[23] B. Mooring and T. Pack, "Determination and specification of robot repeatability," in *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, vol. 3, Apr 1986, pp. 1017–1023.

[24] R. Kluz and T. Trzepieciski, "The repeatability positioning analysis of the industrial robot arm," *Assembly Automation*, vol. 34, no. 3, pp. 285–295, 2014.

[25] J.-F. Brethe and B. Dakyo, "A stochastic ellipsoid approach to repeatability modelisation of industrial manipulator robots," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, vol. 2, 2002, pp. 1608–1613 vol.2.

[26] J.-F. Brethe, E. Vasselin, D. Lefebvre, and B. Dakyo, "Modelling of repeatability phenomena using the stochastic ellipsoid approach," *Robotica*, vol. 24, pp. 477–490, 7 2006.

[27] American National Standard Institute, Inc, *Point-to-Point and Static Performance Characteristics - Evaluation: R15.05-1-1990*, ser. American National Standard for Industrial Robots and Robot Systems, 1990.

[28] J. Jeswiet and R. Helferty, "Measuring robot repeatability an application of iso and ansi standards," *Advanced Robotics*, vol. 10, no. 5, pp. 503–520, 1995.

[29] "Arm Control Overview," 2014. [Online]. Available: http://sdk.rethinkrobotics.com/wiki/Arm_Control_Overview

[30] "Zero-G Mode," 2014. [Online]. Available: http://sdk.rethinkrobotics.com/wiki/Zero-G_Mode

[31] "Mocap_optitrack," 2013. [Online]. Available: http://wiki.ros.org/mocap_optitrack

[32] "File:workspace front.png," 2014. [Online]. Available: http://sdk.rethinkrobotics.com/wiki/File:Workspace_front.png

[33] "Arm Calibration," 2014. [Online]. Available: http://sdk.rethinkrobotics.com/wiki/Calibration