Kinematics and Calibration for a Robot Comprised of Two Barrett WAMs and a Point Grey Bumblebee2 Stereo Camera

Benjamin Balaguer and Stefano Carpin

School of Engineering Technical Report 2012001
University of California, Merced
Merced, CA, USA

August 2012
Chapter 1

Introduction

In this technical report, we describe the steps required to correctly and accurately setup a manipulation robot. More specifically, we describe the guidelines, algorithms, and mathematical equations that are essential for the robot to perform simple tasks such as moving its manipulator to a specific point in space or acquiring three-dimensional data points with a stereo vision system. Although the discussions are specific to our robot (see Figure 1.1), the majority of the algorithms presented can be effortlessly applied to a different robotic system. The robot we use is comprised of two Barrett WAM arms mounted sideways on a rigid robotic torso along with a Point Grey Bumblebee2 stereo camera. The robot has been built to be anthropomorphic by attempting to mimic the upper-body of a human. The stereo camera is used to gather information about the environment, which is exploited by the manipulators to perform various manipulation tasks. The technical report is essentially divided into two sections, with one section focusing on the manipulators and the other on the stereo camera. The manipulator’s kinematics, described in Chapter 2, provide functions that map the manipulator’s joint values to the end-effector’s position and rotation (i.e., forward kinematics) and vice-versa (i.e., inverse kinematics). Although the forward kinematic equations are general enough to apply to any manipulator for which Denavit-Hartenberg (DH) parameters exist, solving the inverse kinematic equations analytically is dependent on the manipulator. The analytical process of solving the inverse kinematic equations described in this technical report can nevertheless be applied to different manipulators, even though the final equations cannot. The stereo camera requires stereo vision to infer three-dimensional information from the environment, a problem that has been solved many different ways in the past. We describe the fundamentals of stereo vision in Chapter 3, giving a high-level overview of the process, describing the various tradeoffs between different algorithmic choices, and highlighting the design choices that work best for manipulation robotic scenarios. With high accuracy being a crucial behavior in robotic manipulation, we describe, in Chapter 4, specific algorithms we designed in order to precisely calibrate both the manipulators and the stereo camera. These calibration algorithms compute a global reference frame shared between all components of the robotic system in such a way that errors in reference frame conversion (e.g., when converting a point in the camera’s local reference frame to a point in the manipulator’s reference frame) are minimized.
Figure 1.1: Picture of the robotic system with the two Barrett WAM arms and a Point Grey Bumblebee2 stereo camera.
Chapter 2

Kinematics

The Barret WAM arm can be arranged in two distinct configurations, depending on whether or not the wrist is attached to the end of the manipulator. The first configuration, the \textit{4-DoF manipulator}, is comprised of four DoFs and does not encompass the wrist. The four DoFs (J1 through J4 in Figure 2.1) are located on the shoulder (3) and at the elbow (1). The second configuration, the \textit{7-DoF manipulator}, includes the wrist in addition to the 4-DoF manipulator, resulting in a total of seven DoFs. The seven DoFs (J1 through J7 in Figure 2.1) are located on the shoulder (3), the elbow (1), and the wrist (3). Evidently, when configured without the wrist, the Barrett WAM becomes a 4-DoF manipulator that has no guarantee of being able to reach a particular location and orientation. Consequently, we exclusively exploit the Barret WAM in its 7-DoF configuration. The 4-DoF configuration is nevertheless important since it can facilitate the analytical solution to the inverse kinematics problem and, as such, will be described in throughout this technical report.

An important design feature of the 7-DoF configuration is that it contains a spherical wrist. A spherical wrist is designed in such a way that all the axes of rotation intersect at a single point. As will be explained in subsequent sections, the spherical wrist will simplify the analytical solution to the inverse kinematics problem, since it allows the end-effector position and orientation to be decoupled and solved separately.

Figure 2.1: Profile (first) and isometric (second) representations of a Barrett WAM arm, configured either as a 4-DoF (left) or 7-DoF (right) manipulator. The \textit{i}th joint is labeled as \(J_i\) and each joint’s coordinate frame labeled as \(X_i\) (red arrow), \(Y_i\) (green arrow), and \(Z_i\) (blue arrow).
2.1 DH Parameters

Solving the kinematics of the Barrett WAM manipulator starts with attaching reference frames to each joint. Although different conventions have been proposed \[1, 2, 3\], the most popular, and the one we use, are the DH parameters \[1, 4, 5\]. The DH parameters consist of 4 parameters for each joint \(i\), \(a_i\), \(\alpha_i\), \(d_i\), and \(\theta_i\) that together define the coordinate frame for joint \(i\) relative to joint \(i-1\). Only four parameters are needed, as opposed to six (i.e., \(X\), \(Y\), \(Z\), roll, pitch, yaw), thanks to the following set of coordinate convention: 1) \(z_i\) dictates the axis of rotation for joint \(i\), 2) \(x_i\) is parallel to the common normal of \(z_i\) and \(z_{i-1}\) and 3) \(y_i\) is perpendicular to both \(x_i\) and \(z_i\), following the right-hand rule. As shown in Figure 2.2, \(a_i\) is the offset along \(X_i\), \(\alpha_i\) is the angle around \(X_{i-1}\) that rotates \(Z_{i-1}\) to \(Z_i\), \(d_i\) is the offset along \(Z_i\), and \(\theta_i\) is angle of rotation around \(Z_i\) (i.e., for revolute joints, \(\theta_i\) is a free parameter dictated by the joint values given to the manipulator). It is worthwhile to note that DH parameters constitute a minimal representation and provides a fundamental mathematical framework for the computation of arm kinematics. In the next two subsections, we provide the specific Barrett WAM manipulator DH parameters for the two possible configurations, which match the coordinate frames shown in Figure 2.1.

![Figure 2.2: Representative example diagram for DH parameters showing two links, \(i-1\) and \(i\), with their corresponding coordinate frames (\(X\), \(Y\), and \(Z\)) and DH parameters (\(a\), \(\alpha\), \(d\), and \(\theta\)).](image)

2.1.1 4-DoF Manipulator

2.1.2 7-Dof Manipulator

2.2 Transformation Matrices

The aforementioned DH parameter methodology allows for a constant and straightforward representation for the kinematic equations of a manipulator. Indeed, the DH parameters can be directly plugged-in a generalized transformation matrix to represent the configuration (i.e., position and orientation) of one joint with respect to another. Consequently, the kinematic
### 2.2.1 Generalized Form

The generalized transformation matrix, which describes a coordinate frame, $i$, relative to its previous coordinate frame, $i-1$, works for any manipulator for which DH parameters have been computed and can describe any coordinate frame. The generalized form is shown below, where the four parameters, $a_i$, $d_i$, $\alpha_i$, and $\theta_i$, refer to the DH parameters for coordinate frame $i$, as shown in Tables 2.1 and 2.2.

$$T_i^{i-1} = \begin{pmatrix}
\cos(\theta_i) & -\sin(\theta_i) \cos(\pi/2) & \sin(\theta_i) \sin(\pi/2) & a_i \cos(\theta_i) \\
\sin(\theta_i) & \cos(\theta_i) \cos(\pi/2) & -\cos(\theta_i) \sin(\pi/2) & a_i \sin(\theta_i) \\
0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\
0 & 0 & 0 & 1
\end{pmatrix}$$

### 2.2.2 Specific Transformation Matrices

Plugging-in the DH-parameters into the generalized transformation matrix, we can come up with the individual transformation matrices for each of the Barrett WAM arm’s coordinate frames. Once each of the coordinate frames have been deduced, they can be used to solve the forward or inverse kinematics problem.

#### 4-DoF Manipulator

There are four degrees of freedom in the 4-DoF configuration for the Barrett WAM arm. We need five transformation matrices, however, so that the end-effector is correctly positioned (otherwise it will be positioned at the fourth joint). We label the last transformation matrix $T_w^4$. In this section, we are exploiting the generalized transformation matrix from Section 2.2.1 by plugging in the DH parameters for the 4-DoF manipulator from Table 2.1. We additionally simplify the transformation matrices.

$$T_1^0 = \begin{pmatrix}
\cos(\theta_1) & -\sin(\theta_1) \cos(-\pi/2) & \sin(\theta_1) \sin(-\pi/2) & 0.00 \\
\sin(\theta_1) & \cos(\theta_1) \cos(-\pi/2) & -\cos(\theta_1) \sin(-\pi/2) & 0.00 \\
0.00 & \sin(-\pi/2) & \cos(-\pi/2) & 0.00 \\
0.00 & 0.00 & 0.00 & 1.00
\end{pmatrix} = \begin{pmatrix}
\cos(\theta_1) & 0.00 & -\sin(\theta_1) & 0.00 \\
\sin(\theta_1) & 0.00 & \cos(\theta_1) & 0.00 \\
0.00 & -1.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 1.00
\end{pmatrix}$$

$$T_2^1 = \begin{pmatrix}
\cos(\theta_2) & -\sin(\theta_2) \cos(\pi/2) & \sin(\theta_2) \sin(\pi/2) & 0.00 \\
\sin(\theta_2) & \cos(\theta_2) \cos(\pi/2) & -\cos(\theta_2) \sin(\pi/2) & 0.00 \\
0.00 & \sin(\pi/2) & \cos(\pi/2) & 0.00 \\
0.00 & 0.00 & 0.00 & 1.00
\end{pmatrix} = \begin{pmatrix}
\cos(\theta_2) & 0.00 & \sin(\theta_2) & 0.00 \\
\sin(\theta_2) & 0.00 & -\cos(\theta_2) & 0.00 \\
0.00 & 1.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 1.00
\end{pmatrix}$$

### Table 2.1: DH Parameters for the 4-DoF manipulator configuration (frames 1 through 4).

<table>
<thead>
<tr>
<th>$i$</th>
<th>$a_i$</th>
<th>$\alpha_i$</th>
<th>$d_i$</th>
<th>$\theta_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00</td>
<td>$-\pi/2$</td>
<td>0.00</td>
<td>$\theta_1$</td>
</tr>
<tr>
<td>2</td>
<td>0.00</td>
<td>$\pi/2$</td>
<td>0.00</td>
<td>$\theta_2$</td>
</tr>
<tr>
<td>3</td>
<td>0.045</td>
<td>$-\pi/2$</td>
<td>0.55</td>
<td>$\theta_3$</td>
</tr>
<tr>
<td>4</td>
<td>-0.045</td>
<td>$\pi/2$</td>
<td>0.00</td>
<td>$\theta_4$</td>
</tr>
<tr>
<td>w</td>
<td>0.00</td>
<td>0.00</td>
<td>0.30</td>
<td>0.00</td>
</tr>
</tbody>
</table>

### Table 2.2: DH Parameters for the 7-DoF manipulator configuration (frames 5 through 7). Frames 1 through 4 are given in Table 2.1.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$a_i$</th>
<th>$\alpha_i$</th>
<th>$d_i$</th>
<th>$\theta_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.00</td>
<td>$-\pi/2$</td>
<td>0.30</td>
<td>$\theta_5$</td>
</tr>
<tr>
<td>6</td>
<td>0.00</td>
<td>$\pi/2$</td>
<td>0.00</td>
<td>$\theta_6$</td>
</tr>
<tr>
<td>7</td>
<td>0.00</td>
<td>0.00</td>
<td>0.06</td>
<td>$\theta_7$</td>
</tr>
<tr>
<td>w</td>
<td>0.00</td>
<td>0.00</td>
<td>0.10</td>
<td>0.00</td>
</tr>
</tbody>
</table>

\[ T_3^2 = \begin{pmatrix} \cos(\theta_3) & -\sin(\theta_3) \cos(-\pi/2) & \sin(\theta_3) \sin(-\pi/2) & 0.045 \cos(\theta_3) \\ \sin(\theta_3) & \cos(\theta_3) \cos(-\pi/2) & -\cos(\theta_3) \sin(-\pi/2) & 0.045 \sin(\theta_3) \\ 0.00 & \sin(-\pi/2) & \cos(-\pi/2) & 0.55 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{pmatrix} \]

\[ T_4^3 = \begin{pmatrix} \cos(\theta_4) & -\sin(\theta_4) \cos(\pi/2) & \sin(\theta_4) \sin(\pi/2) & -0.045 \cos(\theta_3) \\ \sin(\theta_4) & \cos(\theta_4) \cos(\pi/2) & -\cos(\theta_4) \sin(\pi/2) & -0.045 \sin(\theta_3) \\ 0.00 & \sin(\pi/2) & \cos(\pi/2) & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{pmatrix} \]

\[ T_w^4 = \begin{pmatrix} \cos(0) & -\sin(0) \cos(0) & \sin(0) \sin(0) & 0.00 \\ \sin(0) & \cos(0) \cos(0) & -\cos(0) \sin(0) & 0.00 \\ 0.00 & \sin(0) & \cos(0) & 0.30 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{pmatrix} \]

7-DoF Manipulator

There are seven degrees of freedom in the 7-DoF configuration for the Barrett WAM arm. We need eight transformation matrices, however, so that the end-effector is correctly positioned (otherwise it will be positioned at the seventh joint). We label the last transformation matrix \( T_w^7 \). Similarly to the previous section, we are exploiting the generalized transformation matrix from Section 2.2.1 by plugging in the DH parameters for the 7-DoF manipulator from Table 2.2. We only show the transformations for joints 5 through 7 since the transformations for joints 1 through 4 are already given in the previous section.

\[ T_5^4 = \begin{pmatrix} \cos(\theta_5) & -\sin(\theta_5) \cos(-\pi/2) & \sin(\theta_5) \sin(-\pi/2) & 0.00 \\ \sin(\theta_5) & \cos(\theta_5) \cos(-\pi/2) & -\cos(\theta_5) \sin(-\pi/2) & 0.00 \\ 0.00 & \sin(-\pi/2) & \cos(-\pi/2) & 0.30 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{pmatrix} \]

\[ T_6^5 = \begin{pmatrix} \cos(\theta_6) & -\sin(\theta_6) \cos(\pi/2) & \sin(\theta_6) \sin(\pi/2) & 0.00 \\ \sin(\theta_6) & \cos(\theta_6) \cos(\pi/2) & -\cos(\theta_6) \sin(\pi/2) & 0.00 \\ 0.00 & \sin(\pi/2) & \cos(\pi/2) & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{pmatrix} \]

\[ T_7^6 = \begin{pmatrix} \cos(\theta_7) & -\sin(\theta_7) \cos(0.00) & \sin(\theta_7) \sin(0.00) & 0.00 \\ \sin(\theta_7) & \cos(\theta_7) \cos(0.00) & -\cos(\theta_7) \sin(0.00) & 0.00 \\ 0.00 & \sin(0.00) & \cos(0.00) & 0.06 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{pmatrix} \]

\[ T_w^7 = \begin{pmatrix} \cos(0.00) & -\sin(0.00) \cos(0.00) & \sin(0.00) \sin(0.00) & 0.00 \\ \sin(0.00) & \cos(0.00) \cos(0.00) & -\cos(0.00) \sin(0.00) & 0.00 \\ 0.00 & \sin(0.00) & \cos(0.00) & 0.10 \\ 0.00 & 0.00 & 0.00 & 1.00 \end{pmatrix} \]

2.3 Forward Kinematics

The goal of the forward kinematics problem is to compute the location and rotation of a manipulator’s end-effector, given its joint values \( (\theta_1 \ldots \theta_7) \). Although forward kinematics can be used to find the coordinate frame of any joint along the manipulator, it is most often exploited to find the configuration of the end-effector. With the aforementioned transformation matrices, given in Sections 2.2.2 and 2.2.2, it is straightforward to calculate the forward kinematics by multiplying the transformation matrices together up to the frame for which we need its position and orientation. The
result is a transformation matrix, from which we can extract the end-effector’s position and rotation matrix. For example, to get the transformation matrix of frame 3, we would compute
\[ T_3^0 = T_3^1 T_2^1 T_1^2. \]
Similarly and as a second example, to get the transformation matrix of the end-effector for the 7-DoF Barret WAM arm configuration, we would compute the following matrix multiplication:
\[ T_w^0 = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 T_7^6. \]
In this section, we only provide the resulting transformation matrix for the 4-DoF configuration since it will be exploited to calculate the inverse kinematics. It is straightforward, however, to retrieve similar information from the transformation matrices, as exemplified previously.
\[ T_w^0 = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 \]
Due to space constraint, we write the transformation as a rotation matrix and translation vector. They can easily be combined to form a transformation matrix.
\[
R_w^0 = \begin{pmatrix}
   c_1 c_2 c_3 c_4 - s_1 s_3 c_4 - c_1 s_2 s_4 & -c_1 c_2 s_3 - s_1 c_4 & c_1 c_2 c_3 s_4 - s_1 s_3 s_4 + c_1 s_2 c_4 \\
   s_1 c_2 c_3 c_4 + s_1 s_3 c_4 - s_1 s_2 s_4 & -s_1 c_2 s_3 + c_1 c_4 & s_1 c_2 c_3 s_4 + s_1 s_3 s_4 + s_1 s_2 c_4 \\
   -s_2 c_3 c_4 - c_2 s_4 & s_2 s_3 & -s_2 c_3 s_4 + c_2 c_4
\end{pmatrix}
\]
\[
P_w^0 = \begin{pmatrix}
   c_1 c_2 c_3 s_4 d_w - s_1 s_3 s_4 d_w + c_1 s_2 c_4 d_w + c_1 c_2 c_4 a_3 c_4 - s_1 s_3 a_4 c_4 - c_1 s_2 a_4 s_4 + c_1 c_2 a_3 c_3 - s_1 a_3 c_3 + c_1 s_2 d_3 \\
   s_1 c_2 c_3 s_4 d_w + c_1 s_3 s_4 d_w + s_1 s_2 c_4 d_w + s_1 c_2 s_4 d_w + c_1 c_2 s_4 a_4 c_4 + c_1 s_3 a_4 c_4 - c_1 s_2 a_4 s_4 + s_1 a_3 c_3 + c_1 a_3 s_3 + s_1 s_2 d_3 \\
   -s_2 c_3 s_4 d_w + c_2 c_4 d_w - s_2 c_4 a_4 c_4 - c_2 a_4 s_4 - s_2 a_3 c_3 + c_2 d_3
\end{pmatrix}
\]
### 2.4 Inverse Kinematics

Conversely to the forward kinematics problem, the goal of inverse kinematics is to find a set of manipulator joint values \((\theta_1 \ldots \theta_n)\), given an end-effector location and rotation. Effectively, we need to reverse the process performed in forward kinematics. The difficulty in solving the inverse kinematics problem is manipulator-dependent, where simple non-redundant manipulators can generally be solved analytically [6] and more complicated redundant manipulators need to exploit advanced mathematical frameworks. Consequently, and not surprisingly, numerous approaches have been proposed to solve IK problems, such as coordinate descent [7], Jacobian pseudoinverse [8], Jacobian transpose [9], damped least squares [10], conjugate gradient [11], and artificial intelligence [12] methods. All of these complex solutions come with non-negligible increases in time complexity costs, when compared with analytical solutions. The Barrett WAM arm has a redundant DoF, making a direct analytical solution impossible. Without this redundant DoF, however, the arm is an anthropomorphic arm for which an analytical solution exists [6]. Similarly to [13], we solve the inverse kinematics problem by setting the redundant DoF (i.e., joint 3) as a free parameter (i.e., an input variable that has a fixed value when solving for inverse kinematics) and solving the inverse kinematics problem analytically. We note that a geometrical solution for the Barrett WAM arm has been demonstrated in [14] but that it is slower than the analytical method we propose.

The design of the Barrett WAM arm provides an additional simplification when it comes to analytically solve the inverse kinematics problem. The wrist, which accounts for the last three DoFs of the 7-DoF manipulator configuration, is spherical, meaning that all three axes of rotation intersect at a single point. This design feature allows for a decoupling of the inverse kinematics problem, where the end-effector’s position can be solved independently of the end-effector’s rotation. Put differently, the end-effector’s location can be solved by computing inverse kinematics for the 4-DoF Manipulator configuration and the end-effector’s rotation can be solved by computing inverse kinematics for the spherical wrist (i.e., 7-DoF Manipulator configuration). The input to the inverse kinematics solver is a transformation \( T_e \), which can be decomposed in the end-effector’s position, \( p_e = [X_e Y_e Z_e] \), and rotation, \( R_e = [n_e s_e a_e] \) (note that \( R_e \) is a matrix where \( n_e \), \( s_e \), and \( a_e \) are unit vectors).

#### 2.4.1 4-DoF Manipulator

The first component that we solve for the analytical solution of the Barrett WAM arm is the 4-DoF configuration, in which we will focus on the end-effector’s location. We start by computing the point \( p = [X Y Z] \) corresponding to the intersection of the three joints of the spherical wrist using the formula
\[ p = p_e - (d_7 + d_w)a_e \]
where $p_e$ is the end-effector’s desired location, $d_7$ and $d_w$ come from the DH parameters in Table 2.2. and $a_e$ is the unit vector for the end-effector’s desired Z axis rotation (taken from $R_e$). Reformulating the inverse kinematics problem in this context, we are given a position $p = [X Y Z]$ and need to find the possible joint angles (i.e., $\theta_1, \theta_2, \theta_3, \theta_4$) that will drive the end-effector to that position. Using the translation matrix $P^0_w$ (see Section 2.3), we have 3 equations to solve for 4 parameters. Since we have a redundant manipulator, there is an infinite amount of solutions to solve inverse kinematics, a problem that we circumvent, as previously mentioned, by setting a redundant joint as a free parameter. For our solution, we arbitrarily choose joint 3 to be a free parameter (i.e., $\theta_3$), and that we can set it to a value and solve inverse kinematics with that value. Using different values for joint 3, with the same end-effector position, allows us to effectively sample through the arm configuration space with the same end-effector position. Having joint 3 as a free parameter means that we now have three parameters ($\theta_1, \theta_2, \theta_3$) and three equations $(X, Y, Z)$ and that we can solve the inverse kinematics analytically, as shown in the next subsections.

Starting Equations

We start by rewriting the translation matrix $P^0_w$ given in Section 2.3 in terms of $X$, $Y$, and $Z$.

\[
X = c_1 c_2 c_3 s_4 d_w - s_1 s_3 s_4 d_w + c_1 s_2 c_4 d_w + c_1 c_2 c_3 a_4 c_4 - s_1 s_3 a_4 c_4 - c_1 s_2 a_4 c_4 + c_1 c_2 a_3 c_3 - s_1 a_3 s_3 + c_1 s_2 d_3 \quad (2.4.1)
\]

\[
Y = s_1 c_2 c_3 s_4 d_w + c_1 s_3 s_4 d_w + s_1 s_2 c_3 d_w + s_1 c_2 c_3 a_4 c_4 + c_1 s_3 a_4 c_4 - s_1 s_2 a_4 c_4 + s_1 c_2 a_3 c_3 + c_1 a_3 s_3 + s_1 s_2 d_3 \quad (2.4.2)
\]

\[
Z = -s_2 c_3 s_4 d_w + c_2 c_4 d_w - s_2 c_3 a_4 c_4 - c_2 a_4 s_4 - s_2 a_3 c_3 + c_2 d_3 \quad (2.4.3)
\]

Solving for $\theta_4$

We can rewrite $X$ and $Y$, from (2.4.1) and (2.4.2) respectively, by factoring out $c_1$ and $s_1$:

\[
X = c_1 (c_2 c_3 s_4 d_w + s_2 c_4 d_w + c_2 c_3 a_4 c_4 - s_2 a_4 s_4 + c_2 a_3 c_3 - s_2 d_3) - s_1 (s_3 s_4 d_w + s_3 a_4 c_4 + a_3 s_3) \quad (2.4.4)
\]

\[
Y = s_1 (c_2 c_3 s_4 d_w + s_2 c_4 d_w + c_2 c_3 a_4 c_4 - s_2 a_4 s_4 - c_2 a_3 c_3 + s_2 d_3) + c_1 (s_3 s_4 d_w + s_3 a_4 c_4 + a_3 s_3) \quad (2.4.5)
\]

Let:

\[
A = c_2 c_3 s_4 d_w + s_2 c_4 d_w + c_2 c_3 a_4 c_4 - s_2 a_4 s_4 + c_2 a_3 c_3 + s_2 d_3 \quad (2.4.6)
\]

\[
B = s_3 s_4 d_w + s_3 a_4 c_4 + a_3 s_3 \quad (2.4.7)
\]

Such that, plugging in $A$ and $B$ into (2.4.4) and (2.4.5):

\[
X = c_1 A - s_1 B \quad (2.4.8)
\]

\[
Y = s_1 A + c_1 B \quad (2.4.9)
\]

Combining and simplifying the equations:

\[
X^2 + Y^2 = (c_1 A - s_1 B)^2 + (s_1 A + c_1 B)^2
\]
\[
X^2 + Y^2 = c_1^2 A^2 - c_1 s_1 AB + s_1^2 B^2 + s_1^2 A^2 + s_1 c_1 AB + c_1^2 B^2
\]
\[
X^2 + Y^2 = (c_1^2 + s_1^2) A^2 + (c_1^2 + s_1^2) B^2 + (c_1 s_1 - c_1 s_1) AB
\]
\[
X^2 + Y^2 = A^2 + B^2 \quad (2.4.10)
\]

Adding $Z$ on both sides:

\[
X^2 + Y^2 + Z^2 = A^2 + B^2 + Z^2
\]

Substituting $A$, $B$, and $Z$, with (2.4.6), (2.4.7), and (2.4.3), respectively.

\[
X^2 + Y^2 + Z^2 = (c_2 c_3 s_4 d_w + s_2 c_4 d_w + c_2 c_3 a_4 c_4 - s_2 a_4 s_4 + c_2 a_3 c_3 + s_2 d_3)^2
\]
\[
+ (s_3 s_4 d_w + s_3 a_4 c_4 + a_3 s_3)^2
\]
\[
+ (-s_2 c_3 s_4 d_w + c_2 c_4 d_w - s_2 c_3 a_4 c_4 - c_2 a_4 s_4 - s_2 a_3 c_3 + c_2 d_3)^2
\]
Expanding the squares and canceling terms:

\[
X^2 + Y^2 + Z^2 = c_2^2 s_2^2 d_3^2 + 2c_2^2 c_4 s_4 a_4 d_w + 2c_2^2 c_4 s_4 a_3 d_w + c_4^2 s_3^2 a_4 d_w + 2c_2^2 c_4 s_4 a_3 d_w + 2c_4^2 c_4 s_8 s_7 d_2 d_w
\]

Factoring \(c_2^2 + s_2^2\) and simplifying the equation using the trigonometric identity \(c_2^2 + s_2^2 = 1\):

\[
X^2 + Y^2 + Z^2 = c_2^2 s_2^2 d_3^2 + 2c_2^2 c_4 s_4 a_4 d_w + 2c_2^2 c_4 s_4 a_3 d_w + c_4^2 s_3^2 a_4 d_w - 2c_4 a_4 s_4 d_w + 2c_4 s_4 d_3 d_w
\]
Factoring $c_3^2 + s_3^2$ and simplifying the equation using the trigonometric identity $c_3^2 + s_3^2 = 1$:

$$X^2 + Y^2 + Z^2 = s_3^2 d_w^2 + 2c_4 s_4 a_4 d_w + 2c_4 s_4 a_4 d_w + c_4^2 d_w^2 - 2c_4 s_4 a_4 d_w + 2c_4 d_3 d_w$$
$$+ s_3^2 d_w^2 + 2c_3 c_4 a_3 a_4 + s_4^2 a_4 - 2s_4 a_4 d_3 + c_3^2 a_3^2 + d_3^2$$
$$+ s_3^2 d_w^2 + 2c_4 s_3 a_4 d_w + 2c_4 s_3 a_4 d_w + c_3^2 a_3^2 + 2c_4 s_3 a_3 a_4 + s_3^2 a_3^2$$
$$= (s_3^2 d_w^2) (c_3^2 + a_3^2) + (2c_4 s_4 a_4 d_w) (c_3^2 + s_3^2) + (2c_4 s_3 a_4 d_w) (c_3^2 + s_3^2) + c_3^2 d_w^2 - 2c_4 s_4 a_4 d_w + 2c_4 d_3 d_w$$
$$+ (c_3^2 a_3^2) (c_3^2 + a_3^2) + (2c_4 a_3 a_4) (c_3^2 + a_3^2) + s_4^2 a_4^2 - 2s_4 a_4 d_3 + (a_3^2) (c_3^2 + a_3^2) + d_3^2$$
$$= s_3^2 d_w^2 + 2c_4 s_4 a_4 d_w + 2s_4 a_3 d_w + c_3^2 d_w^2 - 2c_4 s_4 a_4 d_w + 2c_4 d_3 d_w$$
$$+ c_4^2 a_4^2 + 2c_4 a_3 a_4 + s_4^2 a_4^2 - 2s_4 a_4 d_3 + a_3^2 + d_3^2$$
$$= s_3^2 d_w^2 + 2s_4 a_3 d_w + c_4^2 d_w^2 + 2c_4 d_3 d_w + c_4^2 a_4^2 + 2c_4 a_3 a_4 + s_4^2 a_4^2 - 2s_4 a_4 d_3 + a_3^2 + d_3^2$$

Factoring $c_3^2 + s_3^2$ and simplifying the equation using the trigonometric identity $c_3^2 + s_3^2 = 1$:

$$X^2 + Y^2 + Z^2 = s_3^2 d_w^2 + 2s_4 a_3 d_w + c_4^2 d_w^2 + 2c_4 d_3 d_w + c_4^2 a_4^2 + 2c_4 a_3 a_4 + s_4^2 a_4^2 - 2s_4 a_4 d_3 + a_3^2 + d_3^2$$
$$= (d_w^2) (c_3^2 + s_3^2) + 2s_4 a_3 d_w + 2c_4 d_3 d_w + (a_3^2) (c_3^2 + s_3^2) + 2c_4 a_3 a_4 - 2s_4 a_4 d_3 + a_3^2 + d_3^2$$
$$= d_w^2 + 2s_4 a_3 d_w + 2c_4 d_3 d_w + a_3^2 + 2c_4 a_3 a_4 - 2s_4 a_4 d_3 + a_3^2 + d_3^2$$

We now have a formula in terms of $\theta_4$ only. As such, we can separate the terms:

$$X^2 + Y^2 + Z^2 = d_w^2 + 2s_4 a_3 d_w + 2c_4 d_3 d_w + a_3^2 + 2c_4 a_3 a_4 - 2s_4 a_4 d_3 + a_3^2 + d_3^2$$

$$X^2 + Y^2 + Z^2 = a_3^2 - a_4^2 - d_3^2 - d_w^2 - 2(s_4 a_3 d_w - s_4 a_4 d_3 + c_4 d_3 d_w + c_4 a_3 a_4)$$

$$\frac{X^2 + Y^2 + Z^2 - a_3^2 - a_4^2 - d_3^2 - d_w^2}{2} = s_4 a_3 d_w - s_4 a_4 d_3 + c_4 d_3 d_w + c_4 a_3 a_4$$

(2.4.11)

Let:

$$N = \frac{X^2 + Y^2 + Z^2 - a_3^2 - a_4^2 - d_3^2 - d_w^2}{2}$$

(2.4.12)

$$R = a_3 d_w - a_4 d_3$$

(2.4.13)

$$T = d_3 d_w + a_3 a_4$$

(2.4.14)

Such that, plugging in $N$ (2.4.12), $R$ (2.4.13), and $T$ (2.4.14) into (2.4.11):

$$N = Rs_4 + Tc_4$$

(2.4.15)

$$N = Rs_4 - Tc_4 = 0$$

(2.4.16)

We need to solve for $\theta_4$, which we achieve by exploiting the following trigonometric identities.

$$\tan \left( \frac{\theta_4}{2} \right) = \pm \sqrt{\frac{1 - \cos(\theta_4)}{1 + \cos(\theta_4)}}$$

(2.4.17)

$$\tan \left( \frac{\theta_4}{2} \right) = \frac{\sin(\theta_4)}{1 + \cos(\theta_4)}$$

(2.4.18)
More specifically, we modify (2.4.16):

\[ N - R s_4 - T c_4 = \left( \frac{N + N - T + T}{2} \right) - R s_4 + \left( \frac{N - T + N - N}{2} \right) c_4 \]

\[ = \left( \frac{N - T}{2} \right) + \left( \frac{N + T}{2} \right) - R s_4 + \left( \frac{N - T}{2} \right) c_4 + \left( \frac{N - N}{2} \right) c_4 \]

\[ = \left( \frac{N - T}{2} \right) + \left( \frac{N + T}{2} \right) - R s_4 + \left( \frac{N - T}{2} \right) c_4 - \left( \frac{N + T}{2} \right) c_4 \]

Dividing by 1 + \( c_4 \) allows us to use the trigonometric identities (2.4.17) and (2.4.18).

\[ \left( \frac{N - T}{2} \right) \left( \frac{1 + c_4}{1 + c_4} \right) + \left( \frac{N + T}{2} \right) \left( \frac{1 - c_4}{1 + c_4} \right) - R \left( \frac{s_4}{1 + c_4} \right) = 0 \]

\[ \left( \frac{N - T}{2} \right) \left( \frac{1}{1 + c_4} \right) + \left( \frac{N + T}{2} \right) \tan^2 \left( \frac{\theta_4}{2} \right) - R \tan \left( \frac{\theta_4}{2} \right) = 0 \]

Let:

\[ a = \left( \frac{N + T}{2} \right) \] (2.4.19)

\[ b = -R \] (2.4.20)

\[ c = \left( \frac{N - T}{2} \right) \] (2.4.21)

Such that, plugging in \( a \) (2.4.19), \( b \) (2.4.20), and \( c \) (2.4.21) gives us a quadratic equation.

\[ a \tan^2 \left( \frac{\theta_4}{2} \right) + b \tan \left( \frac{\theta_4}{2} \right) + c = 0 \] (2.4.22)

The equation can be solved using the quadratic formula.

\[ \tan \left( \frac{\theta_4}{2} \right) = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \]

We can finally solve for \( \theta_4 \), which has two solutions: one when the square root is positive and the other when it is negative.

\[ \theta_{4,I} = 2 \arctan \left( \frac{-b + \sqrt{b^2 - 4ac}}{2a} \right) \] (2.4.23)

\[ \theta_{4,II} = 2 \arctan \left( \frac{-b - \sqrt{b^2 - 4ac}}{2a} \right) \] (2.4.24)

**Solving for \( \theta_2 \)**

We start by solving for \( c_2 \) in the Z equation (2.4.3).

\[ Z = -s_2 c_3 s_4 d_w + c_2 c_4 d_w - s_2 c_3 a_4 c_4 - c_2 a_4 s_4 - s_2 a_3 c_3 + c_2 d_3 \]

\[ Z = (-c_3 s_4 d_w - c_3 a_4 c_4 - a_3 c_3)s_2 + (c_4 d_w - a_4 s_4 + d_3)c_2 \]

\[ c_2 = \frac{Z - (-c_3 s_4 d_w - c_3 a_4 c_4 - a_3 c_3)s_2}{c_4 d_w - a_4 s_4 + d_3} \]

\[ c_2 = \frac{Z + (c_3 s_4 d_w + c_3 a_4 c_4 + a_3 c_3)s_2}{c_4 d_w - a_4 s_4 + d_3} \] (2.4.25)
We use equation (2.4.10) and solve for \( c_2 \) to form a system of two equations with two unknowns. Please note that \( B \) (2.4.7) does not depend on \( c_2 \) or \( s_2 \) and, as such can be treated as a constant. Conversely, \( A \) (2.4.6) does depend on \( c_2 \) and \( s_2 \) and has to be extended.

\[
\begin{align*}
X^2 + Y^2 &= A^2 + B^2 \\
X^2 + Y^2 - B^2 &= A^2 \\
\pm \sqrt{X^2 + Y^2 - B^2} &= A \\
\pm \sqrt{X^2 + Y^2 - B^2} &= c_2 c_3 s_4 d_w + s_2 c_4 d_w + c_2 c_3 a_4 c_4 - s_2 a_4 s_4 + c_2 a_3 c_3 + s_2 d_3 \\
\pm \sqrt{X^2 + Y^2 - B^2} &= (c_3 s_4 d_w + c_3 a_4 c_4 + a_3 c_3) c_2 + (c_4 d_w - a_4 s_4 + d_3) s_2 \\
c_2 &= \frac{\pm \sqrt{X^2 + Y^2 - B^2} - (c_4 d_w - a_4 s_4 + d_3) s_2}{c_3 s_4 d_w + c_3 a_4 c_4 + a_3 c_3} \\
\end{align*}
\] (2.4.26)

Let:

\[
\begin{align*}
M &= c_4 d_w - a_4 s_4 + d_3 \\
P &= c_3 s_4 d_w + c_3 a_4 c_4 + a_3 c_3 \\
\end{align*}
\] (2.4.27) (2.4.28)

Such that, rewriting (2.4.25) and (2.4.26) in terms of \( M \) (2.4.27) and \( P \) (2.4.28) yields

\[
\begin{align*}
c_2 &= \frac{Z + P s_2}{M} \\
c_2 &= \pm \sqrt{X^2 + Y^2 - B^2 - M s_2} \\
\end{align*}
\] (2.4.29) (2.4.30)

We can now set equations (2.4.1) and (2.4.1) equal to each other and solve for \( s_2 \)

\[
\begin{align*}
Z + P s_2 &= \frac{\pm \sqrt{X^2 + Y^2 - B^2} - M s_2}{P} \\
ZP + P^2 s_2 &= \frac{\pm \sqrt{X^2 + Y^2 - B^2} M - M^2 s_2}{P} \\
P^2 s_2 + M^2 s_2 &= \frac{\pm \sqrt{X^2 + Y^2 - B^2} M - ZP}{P^2 + M^2} \\
s_2 &= \frac{\pm \sqrt{X^2 + Y^2 - B^2} M - ZP}{P^2 + M^2} \\
\end{align*}
\] (2.4.31)

Equations and can also be written in terms of \( s_2 \)

\[
\begin{align*}
s_2 &= \frac{Mc_2 - Z}{P} \\
s_2 &= \frac{\pm \sqrt{X^2 + Y^2 - B^2} - Pc_2}{M} \\
\end{align*}
\] (2.4.32) (2.4.33)

We can now set equations (2.4.32) and (2.4.33) equal to each other and solve for \( c_2 \)

\[
\begin{align*}
Z + P s_2 &= \frac{\pm \sqrt{X^2 + Y^2 - B^2} - M s_2}{P} \\
M^2 c_2 - ZM &= \frac{\pm \sqrt{X^2 + Y^2 - B^2} P - P^2 c_2}{P} \\
P^2 c_2 + M^2 c_2 &= \frac{\pm \sqrt{X^2 + Y^2 - B^2} P + ZM}{P^2 + M^2} \\
c_2 &= \frac{\pm \sqrt{X^2 + Y^2 - B^2} P + ZM}{P^2 + M^2} \\
\end{align*}
\] (2.4.34)
To solve for $\theta_2$, we use the trigonometric identity $\tan(\theta_2) = \frac{\sin(\theta_2)}{\cos(\theta_2)}$ along with equations (2.4.34) and (2.4.31).

$$\tan(\theta_2) = \frac{\pm\sqrt{X^2 + Y^2 - B^2 M - ZP}}{\pm\sqrt{X^2 + Y^2 + B^2 P + ZM}}$$

$$\tan(\theta_2) = \frac{\pm\sqrt{X^2 + Y^2 - B^2 M - ZP}}{\pm\sqrt{X^2 + Y^2 + B^2 P + ZM}}$$

Solving for $\theta_2$ yields two solutions, one when the square root is positive and the other when it is negative.

$$\theta_{2, I} = 2 \arctan \left( \frac{\sqrt{X^2 + Y^2 - B^2 M - ZP}}{\sqrt{X^2 + Y^2 + B^2 P + ZM}} \right)$$  \hspace{1cm} (2.4.35)

$$\theta_{2, II} = 2 \arctan \left( \frac{-\sqrt{X^2 + Y^2 - B^2 M - ZP}}{-\sqrt{X^2 + Y^2 + B^2 P + ZM}} \right)$$  \hspace{1cm} (2.4.36)

**Solving for $\theta_1$**

We use (2.4.8) and to solve for $c_1$ and $s_1$, respectively.

$$X = c_1 A - s_1 B$$

$$c_1 = \frac{X + s_1 B}{A}$$  \hspace{1cm} (2.4.37)

$$Y = s_1 A + c_1 B$$

$$s_1 = \frac{Y - c_1 B}{A}$$  \hspace{1cm} (2.4.38)

We plug (2.4.38) into (2.4.37) and solve for $c_1$.

$$c_1 = \frac{X + (Y-c_1 B)B}{A}$$

$$c_1 = \frac{X}{A} + \frac{Y B}{A^2} - \frac{c_1 B^2}{A^2}$$

$$A^2 c_1 = AX + BY - c_1 B^2$$

$$A^2 c_1 + c_1 B^2 = AX + BY$$

$$c_1 = \frac{AX + BY}{A^2 + B^2}$$  \hspace{1cm} (2.4.39)

Similarly, we plug (2.4.37) into (2.4.38) and solve for $s_1$.

$$s_1 = \frac{Y - (X+s_1 B)B}{A}$$

$$s_1 = \frac{Y}{A} - \frac{X B}{A^2} - \frac{s_1 B^2}{A^2}$$

$$A^2 s_1 = AY - BX - s_1 B^2$$

$$A^2 s_1 + s_1 B^2 = AY - BX$$

$$s_1 = \frac{AY - BX}{A^2 + B^2}$$  \hspace{1cm} (2.4.40)
To solve for \( \theta_1 \), we use the trigonometric identity \( \tan(\theta_1) = \frac{\sin(\theta_1)}{\cos(\theta_1)} \) along with equations (2.4.39) and (2.4.40).

\[
\tan(\theta_1) = \frac{\sin(\theta_1)}{\cos(\theta_1)}
\]

\[
\tan(\theta_1) = \frac{AY - BX}{AX + BY}
\]

\[
\tan(\theta_1) = \frac{AY}{AX + BY}
\]

Solving for \( \theta_1 \).

\[
\theta_{1,1} = \arctan \left( \frac{AY - BX}{AX + BY} \right) \tag{2.4.41}
\]

**Final Solutions**

Having analytically solved for \( \theta_4, \theta_2, \) and \( \theta_1 \), we can construct the set of solutions to solve the inverse kinematics problem. It is important to note that the solution for \( \theta_4 \) is independent, the solution for \( \theta_2 \) is dependent on the solutions for \( \theta_4 \), and the solution for \( \theta_1 \) is independent on both the solutions for \( \theta_4 \) and \( \theta_2 \). Consequently, the two solutions for \( \theta_4 \), (2.4.23) and (2.4.24), are combined with the two solutions for \( \theta_2 \), (2.4.35) and (2.4.36), and the solution for \( \theta_1 \), (2.4.41). In other words, given an X, Y, Z position and a parameter for \( \theta_4 \), the inverse kinematics solver produces a set of four solutions, as follows:

\[
\begin{align*}
[\theta_{1,1}, \theta_{1,II}] \\
[\theta_{1,II}, \theta_{1,1}] \\
[\theta_{1,1}, \theta_{2,1}] \\
[\theta_{1,II}, \theta_{2,1}]
\end{align*}
\]

2.4.2 7-DoF Manipulator

In the previous section, we have solved the inverse kinematics problem only taking into account the end-effector’s position and found four solutions for three joint values (i.e., \( \theta_1, \theta_2, \theta_4 \)) assuming that one joint value is already set (i.e., \( \theta_3 \)). We now use the results of the previous section along with the spherical wrist to find the last three joint values of the 7-DoF manipulator (i.e., \( \theta_5, \theta_6, \theta_7 \)). Before we can solve the inverse kinematics problem for the spherical wrist, we must find the wrist orientation relative to the 4-DoF manipulator. Indeed, the joint values for the wrist are dependent on the rotation of the remaining parts of the manipulator. We start by finding the rotation of the last joint (i.e., joint 4) of the 4-DoF configuration, which can easily be acquired using the forward kinematic equation

\[
T_4^0 = T_4^0 T_2^1 T_3^2 T_4^3.
\]

From the transformation \( T_4^0 \), which was calculated using the joint values acquired in the previous section (i.e., \( \theta_1, \theta_2, \theta_3, \) and \( \theta_4 \)), we can extract the 4-DoF Manipulator’s orientation \( R_4^0 \). We can then compute the relative difference between the 4-DoF Manipulator’s orientation \( R_4^0 \) with the desired end-effector’s rotation \( R_e \):

\[
R = (R_4^0)^T R_e.
\]

The inverse kinematics problem for the spherical wrist consists in finding the three joint values (i.e., \( \theta_5, \theta_6, \theta_7 \)) that allow the wrist to be at rotation \( R \). We can rewrite the rotation matrix in terms of rotations around the Z, Y, and Z axes:

\[
R = R_z(\theta_5)R_y(\theta_6)R_z(\theta_7)
\]

\[
= \begin{pmatrix}
c_{\theta_6}c_{\theta_7} - s_{\theta_6}c_{\theta_7} & -c_{\theta_6}c_{\theta_7} - s_{\theta_6}c_{\theta_7} & s_{\theta_6}s_{\theta_7} \\
-c_{\theta_6}c_{\theta_7} + s_{\theta_6}c_{\theta_7} & s_{\theta_6}c_{\theta_7} + c_{\theta_6}c_{\theta_7} & c_{\theta_6}s_{\theta_7} \\
-s_{\theta_6}c_{\theta_7} & -s_{\theta_6}s_{\theta_7} & c_{\theta_6}
\end{pmatrix}
= \begin{pmatrix}
r_{11} & r_{12} & r_{13} \\
r_{21} & r_{22} & r_{23} \\
r_{31} & r_{32} & r_{33}
\end{pmatrix}
\]

We can then solve for each joint value, using trigonometric functions and assuming that the values \( r_{11}, \ldots, r_{13} \) are known.
Solving for $\theta_5$

We can solve for $\theta_5$ by using the tangent trigonometric function, along with $r_{23}$ and $r_{13}$:

$$\frac{r_{23}}{r_{13}} = \frac{\sin \theta_5 \sin \theta_6}{\cos \theta_5 \sin \theta_6} = \frac{\sin \theta_5}{\cos \theta_5} = \tan \theta_5.$$  

The inverse tangent can then be used to deduce the joint value:

$$\theta_{5,l} = \arctan 2(r_{23}, r_{13})$$

Solving for $\theta_6$

We can solve for $\theta_6$ by using the tangent trigonometric function, the Pythagorean identity, and $r_{23}, r_{13}$, and $r_{33}$:

$$\frac{\sqrt{(r_{13})^2 + (r_{23})^2}}{r_{33}} = \frac{\sqrt{\cos \theta_6^2 \sin \theta_6^2 + \sin \theta_6^2 \sin \theta_6^2}}{\cos \theta_6} = \frac{\sqrt{\sin \theta_6^2 (\cos \theta_6^2 + \sin \theta_6^2)}}{\cos \theta_6} = \frac{\sqrt{\sin \theta_6^2}}{\cos \theta_6} = \tan \theta_6$$

The inverse tangent can then be used to deduce the joint value:

$$\theta_{6,l} = \arctan 2(\sqrt{(r_{13})^2 + (r_{23})^2}, r_{33})$$

Solving for $\theta_7$

We can solve for $\theta_7$ by using the tangent trigonometric function, along with $r_{32}$ and $r_{31}$:

$$\frac{r_{32}}{-r_{31}} = \frac{\sin \theta_6 \sin \theta_7}{\sin \theta_6 \cos \theta_7} = \frac{\sin \theta_7}{\cos \theta_7} = \tan \theta_7.$$  

The inverse tangent can then be used to deduce the joint value:

$$\theta_{7,l} = \arctan 2(r_{32}, -r_{31})$$

Final Solutions

The solutions proposed assume that $r_{13} \neq 0$, $r_{23} \neq 0$, and $0 < \theta_6 < \pi$. Since $\theta_6$ should encompass the full $2\pi$ spectrum of rotations, another set of solutions can be defined as follows (these solutions assume that $-\pi < \theta_6 < 0$):

$$\theta_{5,II} = \arctan 2(-r_{23}, -r_{13}) \theta_{6,II} = \arctan 2(-\sqrt{(r_{13})^2 + (r_{23})^2}, r_{33}) \theta_{7,II} = \arctan 2(-r_{32}, r_{31}).$$

Combining the joint values for the 4-DoF Manipulator with those found in this section, we have a total of eight solutions for a given free parameter $\theta_3$. Evidently, more solutions can be acquired for the same end-effector location $p_e$ and rotation $R_e$ by uniformly sampling the free parameter $\theta_3$ and performing the aforementioned calculations. The eight inverse kinematics solutions are as follows:

$$[\theta_{1,l}, \theta_{2,l}, \theta_3, \theta_{4,l}, \theta_{5,l}, \theta_{6,l}, \theta_{7,l}]$$
$$[\theta_{1,II}, \theta_{2,II}, \theta_3, \theta_{4,II}, \theta_{5,II}, \theta_{6,II}, \theta_{7,II}]$$
$$[\theta_{1,l}, \theta_{2,l}, \theta_3, \theta_{4,l}, \theta_{5,l}, \theta_{6,l}, \theta_{7,l}]$$
$$[\theta_{1,II}, \theta_{2,II}, \theta_3, \theta_{4,II}, \theta_{5,II}, \theta_{6,II}, \theta_{7,l}]$$
$$[\theta_{1,l}, \theta_{2,l}, \theta_3, \theta_{4,l}, \theta_{5,l}, \theta_{6,l}, \theta_{7,l}]$$
$$[\theta_{1,II}, \theta_{2,II}, \theta_3, \theta_{4,II}, \theta_{5,II}, \theta_{6,l}, \theta_{7,l}]$$
$$[\theta_{1,l}, \theta_{2,l}, \theta_3, \theta_{4,l}, \theta_{5,l}, \theta_{6,l}, \theta_{7,l}]$$
$$[\theta_{1,II}, \theta_{2,II}, \theta_3, \theta_{4,II}, \theta_{5,II}, \theta_{6,II}, \theta_{7,l}].$$
Chapter 3

Stereo Vision

3.1 Overview

In this chapter, we describe the specific process and algorithms exploited as part of our vision system, which is comprised of a single Point Grey Bumblebee2 stereo camera as shown in Figure 3.1. Since a single camera can only acquire two-dimensional, stereo vision utilizes two cameras looking at the same scene to infer depth and extract three-dimensional information. In some sense, stereo vision attempts to reproduce human vision that is performed using two eyes. With two cameras, as opposed to a single one, we have access to the same scene taken from two different vantage points. These two vantage points, gathered in the form of pictures, can be used to triangulate three-dimensional coordinates from the scene. The two images can be analyzed to find points that are the same in both images. The relative position of these points (i.e., disparity) can then be exploited to extract depth information along with three-dimensional coordinates in the camera’s local reference frame. Put differently, the same point in space was observed by two different cameras. By computing the two three-dimensional lines starting at each camera’s center and ending at the matching point, we can infer the point’s three-dimensional coordinate. In order to compute the three-dimensional lines, a set of camera parameters (e.g., fundamental matrix, focal length, position of image center, skew factor, lens distortion) need to be determined using a calibration process. We do not need to perform such a calibration step to since the cameras’ parameters have been already been computed and stored inside the camera’s firmware by the manufacturer. Computer stereo vision is a popular topic that has been extensively studied over the years and our stereo vision framework employs these techniques. Consequently, this chapter is brief and included for the purpose of completeness rather than novelty. In addition, Interested readers looking for more details on stereo vision and calibration are directed to [15, 16, 17, 18].

Figure 3.1: Point Grey Bumblebee2 stereo camera used as part of our robotic system.
3.2 Mathematical Framework

A typical stereo vision process, which is highlighted in Figure 3.2, is composed of 3 components. First, the images need to be rectified in order to lie in the same plane. Second, matching points in each image have to be determined. Third, the three-dimensional coordinates of the matching points can be inferred. The rectification process corrects image distortion by transforming the images into the same coordinate frame, where the two images have been row-aligned. In other words, once rectification has been performed, the difference between the images can only be horizontal (i.e., along the columns). The rectification process essentially reduces the search space of matching points from two dimensions down to one dimension. Although the specific image rectification process is beyond the scope of this technical report, algorithms have been proposed for planar [19], cylindrical [20], and polar [21] rectification. We note, however, that this process is simplified, once again, by the fact that the Bumblebee2 camera is calibrated and that the essential matrix [22] provides the relationship between corresponding points in each camera image. A sample image rectification process is shown for the left camera in Figures 3.2(a) (distorted) and 3.2(c) (rectified) and for the right camera in Figures 3.2(b) (distorted) and 3.2(d) (rectified). By looking carefully at the left and right rectified images (i.e., Figures 3.2(c) and 3.2(d)), one should notice that they only differ horizontally. Once the images have been rectified and row-aligned, we need to find pixels in each image that represent the same points in space.

As already mentioned, the image rectification process has made the problem of matching pixels easier, since we only need to search one dimension (i.e., along the rows), as opposed to two dimensions (i.e., along the rows and columns). Evidently, since this is a fundamental problem in stereo vision, a lot of algorithms have been proposed, the most popular of which being block matching [23], hierarchical block matching [24], graph cuts [25], and feature-based matching [26]. These algorithms rely on cost functions to decide the similarity between two points, such as the absolute difference, the sum of absolute differences, the normalized cross correlation, the census cost function, just to name a few (see ?? for more information and references on cost functions). Choosing the right method encompasses a couple of tradeoffs that have to be taken into account. The more robust and accurate the method, the slower and sparser the results. For example, hierarchical block matching and graph cuts provide more accurate results with less noisy data points than the standard block matching algorithm but are much slower and generally cannot run in real-time. Similarly, feature-based algorithms tend to be more accurate but produce sparser disparity maps than block-matching algorithms. After empirically trying various methods under conditions similar to what the robot would be confronted with, we have found, as corroborated by the Bumblebee2’s manufacturer [27], that the standard block matching algorithm with a cost function using the sum of absolute differences provided the best results. Unlike the other algorithms, the block matching algorithm runs in real-time (4 frames per second for a 640 × 480 image resolution) while producing disparity maps that are not only dense but also accurate. Mathematically, points in both images are matched using the following minimization function,

\[
\arg \min_{d=d_{\min}} \sum_{i=-\frac{m}{2}}^{\frac{m}{2}} \sum_{j=-\frac{r}{2}}^{\frac{r}{2}} |I_l(y+i, x+j) - I_l(y+i, x+j+d)|,
\]

where \(d_{\min}, d_{\max}, m, I_r, \) and \(I_l\) are the minimum possible disparity, maximum possible disparities, mask size, right image, and left image, respectively. We note that the variables \(d_{\min}, d_{\max}, \) and \(m\) are parameters set by the user to constrain the search space, ultimately increasing the speed of the algorithm. An example resulting disparity image is shown in Figure 3.2(c). It is worthwhile to note that once the disparity map is acquired, validation is performed in order to remove inappropriate matches. This validation step is performed with different processes, such as validating texture (i.e., rejecting points that do not have enough contrast or texture), uniqueness (i.e., rejecting points that are too similar to too many other points), and surface (i.e., rejecting points that create spikes in surfaces). These validation processes are influenced by a set of parameters and thresholds that are manually picked based on the expected conditions of the stereo vision. In addition, they exhibit a tradeoff between accuracy and speed, since they will yield better disparity images but will make the overall algorithm slower.

Once the disparity map has been created, we can straightforwardly find the three-dimensional coordinates, in the camera’s local reference frame, for any pixel for which a disparity value has been deduced (i.e., pixels that were matched in the two images). For pixels that could not be matched in both images, a disparity value does not exist and, consequently, a three-dimensional coordinate cannot be computed with the mathematical formulas below. Assuming that the surrounding neighbors are dense enough, we can, however, perform some interpolation on the disparity map [28] or the point cloud [29] to. We have found our disparity maps to be dense enough not to necessitate such interpolation algorithms, but interested readers should check [28] and [29] for more information. The \(X_i, Y_i, \) and \(Z_i\) coordinate of pixel \(i\) is computed with

\[
Z_i = \frac{fB}{d} X_i = \frac{(R_i - R_c)Z_i}{f} Y_i = \frac{(C_i - C_c)Z_i}{f}
\]
where $f$, $B$, $d$, $R_i$, $C_i$, and $C_c$, are the camera’s focal length, baseline, disparity, row for pixel $i$, row of the image center, column for pixel $i$, and column of the image center, respectively. The parameters $f$, $B$, $R_c$, and $C_c$ need to be computed using a calibration step, which has, in our case, already been achieved by the manufacturer of the Bumblebee2 stereo camera. Performing these computations on the entire disparity essentially converts the disparity map into a three-dimensional point cloud of the scene, which itself can be exploited by the Barrett WAM arms for manipulation tasks.

![Figure 3.2](image1)

**Figure 3.2**: Example of the stereo vision process, starting with the raw left (Figure 3.2(a)) and right (Figure 3.2(b)) images that are rectified in Figures 3.2(c) and 3.2(d), respectively. The disparity map, acquired by the point matching algorithm is shown in Figure 3.2(e).

### 3.3 Accuracy Considerations

Considering that stereo vision will be used for a robotic system that will need precise measurements within its environment, it is of utmost importance to analyze the accuracy of the system along with the reasons for errors. The overall accuracy of a stereo vision system can be attributed to three potential sources of error: the accuracy of the camera parameters, the
proper selection of algorithmic variables, and the environment from which the stereo vision attempts to extract information. First and foremost, the two cameras need to be correctly calibrated so that the necessary camera parameters (i.e., focal length, baseline, image center, essential or fundamental matrix) are accurate. Evidently, if the camera parameters are not calibrated properly and inaccurate with respect to their true values, the point clouds created from the stereo vision process will be inaccurate since the mathematical equations use the camera parameters directly. This is the main benefit in buying a stereo camera such as the Bumblebee2 since the manufacturer will build setup the cameras under rigorous engineering standards and will calibrate them very accurately. More specifically, the manufacturer exploits a calibration process that attempts to minimize the Root-Mean-Square (RMS) pixel error between observed and predicted locations for specific calibration points. In [30], the manufacturer states that the Bumblebee2 stereo camera is usually within 0.08 RMS pixel error, a sign of high calibration since it also encompasses error from the point matching algorithm. The second source of error comes from wrong parameters set in the stereo processing algorithms. Indeed, parameters such as the image resolution, the minimum disparity, the maximum disparity, the mask, and the various validation thresholds can have a direct effect on the accuracy of the system. For example, if the maximum disparity is set to a low number (i.e., 8 pixels) and an object of interest is very close to the camera, the stereo process will yield very bad results because the object’s shift relative to the two cameras will be larger than the maximum disparity. Consequently, great care has to be taken when setting up the algorithmic parameters. Fortunately, robots charged with manipulation tasks will generally analyze scenes from similar distances (related to the manipulators’ wingspan) and a single set of parameters can easily be tuned for most robotic applications. For highly dynamic environments, however, the parameters would need to be modified on the fly. We have found, as is suggested by the manufacturer, that displaying the disparity image and tweaking the parameters online works very well and provides accurate results. Based on a series of informal experiments, we have found a stereo vision accuracy of between one and 2 mm for typical robotic scenarios (i.e., camera is looking at a scene from about one meter away). These results are corroborated by the manufacturer [31]. Finally, the environment can itself lower the accuracy of the system when the images lack texture, have repetitive patterns, or are exposed to unusually high or low light conditions. These problems due to the environment can be very difficult, if not impossible, to fix although some solutions have been proposed such as utilizing structured light to create texture on otherwise texture-less objects [32]. However, these extreme environment conditions are rarely seen when performing typical robot tasks in human environments.
Chapter 4

Calibration

4.1 Introduction

The discussions and mathematical equations presented in the previous sections all assume that the components are operating in their local reference frame. Operating in each component’s reference frame is not convenient, however, since three components (i.e., two Barrett WAM arms and a Bumblebee2 stereo camera) will be using different reference frames. In order to circumvent this issue, and as is done for most, if not all, robotic manipulators, we define a global reference frame which is used for all our operations. We note that the global reference frame can be defined as any point in space with any rotation as long as all of the components can exploit it. Consequently, the global reference frame is arbitrary. Since all the components inherently produce or require data in their local reference frames, it is necessary to compute the various transformations that take a point or a rotation in global coordinates and modifies it to the same point or rotation in the component’s local coordinates. Although these transformations can approximately be inferred, it is of crucial importance that they be as accurate as possible. Indeed, a slight error in the transformation (especially in the rotation) will have a major negative consequence in the manipulators’ accuracy since it will propagate through the kinematic links of the manipulator. In other words, the further the end-effector will be from the manipulator’s local reference frame, the worst the end-effector error. In order to find appropriate and accurate transformation matrices, for the manipulators and the stereo camera, we devise calibration procedures for the manipulators and the stereo camera, presented in the next sections, that exploit a motion capture system manufactured by Vicon. The motion capture, comprised of six infrared cameras (see Figure 4.1), is capable of tracking reflective markers with sub-millimeter accuracy, guaranteeing the accuracy of the calibration. For simplicity, we assume that the arbitrary global reference frame is the same as the motion capture’s reference frame (i.e., the motion capture reference frame can be changed arbitrarily and should be set to what we want the global reference frame to be).

Figure 4.1: Three of the six cameras used for the Vicon motion capture.
4.2 Barrett WAM Arm

4.2.1 Overview

As can be seen from Figure 1.1, the Barrett WAM arms are mounted sideways on a steel frame, making the calculation of the global to local transformation difficult to obtain. More specifically, the challenging aspect of determining the transformation comes from the fact that the local reference frame’s location is inside the manipulator’s shoulder and that, as such, we cannot easily measure it with the motion capture system or by other means. However, we can exploit the revolute behavior of the joints with the motion capture by a process highlighted in Figure 4.2. Specifically, by placing a marker on the outer edge of specific joints, as shown in Figure 4.3, the motion capture system can track it as the joint is rotating. The markers’ location corresponding to different joint rotations will encompass points on a two-dimensional circle, which can be estimated. The Z-axis (i.e., the axis of rotation) of the local reference frame will correspond to the normal of the plane on which the circle’s lie. The X-axis of the local reference frame can then be computed by performing a similar computation on a different joint whose axis of rotation is co-linear with the Z-axis. The second joint can also be exploited to determine the location of the local reference frame by computing the intersection of the two lines that follow each joint’s axis of rotation.

Figure 4.2: Graphical overview of the Barrett WAM calibration process with the motion capture system. The data presented in this figure, reflect real data acquired with the Vicon motion capture for one of our Barrett WAM arm. The variables in the diagram match the ones used in the mathematical derivation.

Although the process we describe is evidently based on our experience with our specific hardware (i.e., Barrett WAM arms and Vicon motion capture), it straightforwardly extends and applies to any robot under any environment, assuming that the same type of data can be extracted (i.e., points in the global reference frame). Consequently, a motion capture system is not necessary and can be replaced by other sensors, although it provides very efficient and accurate readings. It is worthwhile to note that the higher the accuracy of the markers’ location, the better the final calibration.

4.2.2 Mathematical Framework

In this section, we will derive a transformation $T^l_g$ that transforms a point, rotation, or transformation in local coordinate frame into a point, rotation, or transformation in global coordinate frame. We note that, in general, we are interested in the other transformation, $T^g_l$, that transforms a point, rotation, or transformation in global coordinate frame into a point, rotation, or transformation in local coordinate frame. Once $T^l_g$ has been derived, $T^g_l$ is straightforwardly deduced by setting its rotation component to $R^g_l = (R^l_g)^T$ and its position component to $P^g_l = -(R^l_g)^TP^l_g$. We assume that a set of data points has been acquired by placing a marker on the first joint of the manipulator (see Figure 4.3), moving the manipulator, and recording the marker’s three-dimensional location, in the global reference frame, for different joint rotations. Mathematically, we have a vector $J = [j_1, j_2, j_3, \ldots, j_n]$, comprised of $n$ data points in three-dimensional global coordinates $j_i = [X_{j_i}, Y_{j_i}, Z_{j_i}]^T$. Although only three data points (i.e., $n = 3$) are theoretically required for this process
to work, it is evident that the more data points, the better the accuracy of the final solution. The number of data points required for a good solution will also depend on the accuracy of the system that acquires the markers’ location. In the case of our motion capture system, which is accurate to within a millimeter, 10 data points are sufficient (i.e., \( n = 10 \)), but considerably more will be needed for systems that are less accurate. We have an additional similarly-acquired data set, \( K = [k_1, k_2, k_3, \ldots, k_n] \), that corresponds to points around another joint of the manipulator. The axis of rotation for this other joint needs to match the X-axis for the manipulator’s local reference frame. Specifically for the Barrett WAM arm, this constraint is achieved by using joint 3 and rotating joint 2 by exactly 90 degrees. The first data set will be used to compute the local reference frame’s position and Z-axis, whereas the second data set will be used to compute the local reference frame’s X-axis. Since the data points inherently lie in a two-dimensional circle, we first start by finding the best-fit plane for the points. This is achieved by performing least squares on the data points. We start with the mathematical equation for a plane:

\[
A \times X_{j_i} + B \times Y_{j_i} + C \times Z_{j_i} = 1.
\]

From this equation, we can create a minimization problem:

\[
\arg \min_{A,B,C} \sum_{i=0}^{n} A \times X_{j_i} + B \times Y_{j_i} + C \times Z_{j_i} - 1.
\]

We can write the minimization problem in terms of linear algebra by requiring \( J^T J_z = y \), where \( J_z = [ABC]^T \) and \( y = [1, 1, 1, \ldots, 1]^T \). This is a typical linear system, often expressed as \( Ax = y \), that can be solved using the linear least squares closed-form solution that relies on the pseudo-inverse of \( A \) (or, in our case, \( J^T \)). Consequently, the solution to our system is

\[
J_z = ((JJ^T)^{-1}J)y.
\]

In cases where the data points are noisy or potentially unreliable (i.e., inaccuracy in the markers’ location), this process can be encompassed inside the RANSAC [33] method, which will determine the subset of datapoints that best fit the two-dimensional plane. We note that the least square solution to the plane, \( J_z = [ABC]^T \), encompasses the plane’s normal vector, which also represents, when normalized to a unit vector, the Z-axis of the local reference frame. From this point on, when we refer to \( J_z \), we assume that it has been normalized and is a unit vector. By performing the same least square computation on the data points acquired for the second joint, we can similarly calculate the X-axis of the local reference frame:

\[
K_z = \hat{J}_x = ((KK^T)^{-1}K)y.
\]

Since the two data sets \( J \) and \( K \) were taken independently of each other, there is no guarantee that \( J_z \) and \( \hat{J}_x \) are orthogonal, which is a requirement for the final transformation matrix \( T_{gl} \). In order to guarantee orthogonality between the two vectors, we project \( \hat{J}_x \) into the plane whose normal is \( J_z \):

\[
J_x = \hat{J}_x - (\hat{J}_x \cdot J_z)J_z.
\]
Similarly to \( J_z \), \( J_x \) is normalized in such a way that it becomes a unit vector. Once two axes of the local reference frame (i.e., \( J_z \) and \( J_x \)) have been calculated, computing the last axis (i.e., \( J_y \)) can be achieved using the cross product:

\[
J_y = J_x \times J_z.
\]

Together, the three unit vectors \( J_x, J_y, J_z \) dictate the rotational component of the transformation matrix \( T'_y \) (i.e., \( R'_y = [J_x, J_y, J_z] \)). Since we have now computed the rotational component of the transformation matrix, we turn our attention to the position component, \( P'_y \). We find the position of the local reference frame with respect to the global reference frame by fitting circles to the data points \( J \) and \( K \), the center of which will be exploited to determine the position component. Since the circle will lie in two-dimensional space, we transform the data points to the X-Y plane (i.e., only the x and y coordinates will be meaningful, with the z value being the same for every data point). The rotation matrix that to rotates the data points into the X-Y plane is acquired using QR decomposition, which will find a \( Q_j \) and \( R_j \) such that

\[
J_z = Q_j R_j,
\]

where \( Q_j \) is the rotation matrix. Consequently, we can rotate the data points as follows:

\[
J' = Q_j J.
\]

Although the rotated data points \( J' \) are still in three dimensions, the first coordinate, \( F_{x,j} \), can be removed since it will be the same (or close to the same) for every point in the data. From now on, when we refer to \( J' \), we assume that each data point is two-dimensional. Since the data points are now two-dimensional, we can now fit the points to a two-dimensional circle. Similarly to the plane fitting, we start with the mathematical equation for a two-dimensional circle,

\[
(x_j - \hat{C}_x^j)^2 + (y_j - \hat{C}_y^j)^2 = (R_j)^2,
\]

where \( \hat{C}_x^j \) is the X coordinate of the circle’s location, \( \hat{C}_y^j \) is the Y coordinate of the circle’s location, and \( R_j \) is the circle’s radius. We manipulate this equation to write it as a linear system of equations:

\[
(X_{j_i})^2 - 2X_{j_i}\hat{C}_x^j + (\hat{C}_x^j)^2 + (Y_{j_i})^2 - 2Y_{j_i}\hat{C}_y^j + (\hat{C}_y^j)^2 = (R_j)^2 - (\hat{C}_x^j)^2 - (\hat{C}_y^j)^2 + (R_j)^2.
\]

Let \( A = 2\hat{C}_x^j, B = 2\hat{C}_y^j, \) and \( C = (R_j)^2 - (\hat{C}_x^j)^2 - (\hat{C}_y^j)^2 \) so that the equation can be rewritten as

\[
(X_{j_i})^2 + (Y_{j_i})^2 = AX_{j_i} + BY_{j_i} + C
\]

From this equation, we can create a minimization problem:

\[
\arg \min_{A,B,C} \sum_{i=0}^{n} AX_{j_i} + BY_{j_i} + C - (X_{j_i})^2 - (Y_{j_i})^2.
\]

We can write the minimization problem in terms of linear algebra by requiring \( J^T S_j = y \), where \( S_j = [ABC]^T \) and \( y = [(X_{j_1})^2 + (Y_{j_1})^2, (X_{j_2})^2 + (Y_{j_2})^2, ..., (X_{j_n})^2 + (Y_{j_n})^2]^T \). This is a typical linear system, often expressed as \( Ax = y \), that can be solved using the linear least squares closed-form solution that relies on the pseudo-inverse of \( A \) (or, in our case, \( J^T \)). Consequently, the solution to our system is

\[
S_j = [ABC]^T = ((JJ^T)^{-1}J)y.
\]

It is then straightforward to recover the X coordinate of the circle’s location, \( \hat{C}_x^j = A/2 \), the Y coordinate of the circle’s location, \( \hat{C}_y^j = B/2 \), and the circle’s radius, \( R_j = \sqrt{C + (\hat{C}_x^j)^2 + (\hat{C}_y^j)^2} \). Similarly to the plane fitting process, care has to be taken in cases where the data points are noisy or potentially unreliable (i.e., inaccuracy in the markers’ location). Once again, RANSAC can be exploited to determine the subset of datapoints that best fit the two-dimensional circle. We note that, for best results, RANSAC should run on both the plane and circle fitting processes (as opposed to the greedy method of running RANSAC independently and separately on each fitting process) so that the best set of data points, in terms of best fit for both fitting processes combined, are deduced. We now need to back-project the circle’s center into its original three-dimensional coordinates using the rotation matrix computed from the QR decomposition. Since the rotation matrix expects the three coordinates, we have to re-introduce \( F_{x,j} \), which was removed in the circle fitting process.
However, since $F_j$ will not be exactly the same for all $i$ (unless we have a perfect fit), we use the mean across all data points instead, $\overline{F_{ji}}$, as follows:

$$J_C = (Q_j)^T[F_{ji}^{-1}, \hat{C}_z^j, \hat{C}_y^j]^T.$$ 

We have calculated the center of the circle, $J_C$, for the set of data points $J$. The same set of operations can be applied to the second data set, $K$, providing the circle’s center for that joint, $K_C$. It is worthwhile to note that we cannot directly use the circle’s center of the first joint, $J_C$, since it is dictated by the marker’s position and might not coincide with the location of the local reference frame. However, the computations performed with the two data sets allow us to derive the location of the local reference frame. Since the axis of rotation of both joints intersect at a single point, the location of the local reference frame can be computed by the intersection of the two lines along the direction of the $Z$ axes of each joint. In terms of variables, we have two lines with equations

$$L_1 = J_C + sJ_z \text{ and } L_2 = K_C + tK_z,$$

where $s$ and $t$ are scalar variables dictating the position along the line. We note that due to the inevitable noise in data acquisition, the lines will not intersect. In other words, the lines are skewed and we are consequently interested in finding the points, along $L_1$ and $L_2$, that minimize the points’ distances from each other. Although there exists some closed-form solutions based on geometry [34] or distance equation solving [35], we use a least squares solution to not only stay consistent with the rest of the technical report but also since it is, in our opinion, easier to understand. We note, however, that we have implemented the closed-form solutions and that they produce exactly the same results as the least squares method. Given the two lines $L1$ and $L2$, we want to find $s$ and $t$ that will provide the same point

$$J_C + sJ_z = K_C + tK_z.$$ 

Evidently, and as aforementioned, it is extremely unlikely to find $s$ and $t$ that will solve this equation since the lines are likely not to intersect. We can nonetheless exploit this equation and solve it as a least squares problem. Specifically, we need to solve the following minimization problem:

$$\arg\min_{s,t} J_C + sJ_z - K_C - tK_z.$$ 

Since it might not be clear how a linear least square method can be applied to this minimization problem, we can rewrite the equality by isolating the variables from the constants:

$$sJ_z - tK_z = K_C - J_C.$$ 

By letting $A = [J_z, K_z]$, $x = [s, -t]^T$ and $y = K_C - J_C$, we have transformed the problem in the typical linear system $Ax = y$. As shown multiple times in this technical report, the system of equation can be solved with the closed-form solution that relies on the pseudo-inverse of $A$:

$$x = [s, -t]^T = ((A^TA)^{-1}X^T)y.$$ 

Plugging the derived $s$ and $t$ variables into the line equations $L1$ and $L2$ provides the two points, $P1$ and $P2$, that are on each line and between which the distance is smallest. It is worthwhile to note that with our motion capture system and the actual data acquired for our calibration, the two points are only 0.23mm away from each other. This result is a testament to the quality of our calibration process. We finally choose the location of the local reference frame as the middle point between $P1$ and $P2$:

$$P_l^9 = P1 + \frac{P2 - P1}{2}.$$ 

Since we have now computed both $P_l^9$ and $R_l^9$, we can straightforwardly combine them together to deduce the final transformation matrix, $T_l^9$.

### 4.3 Bumblebee2 Camera

#### 4.3.1 Overview

"In terms of testing and verifying the quality of the disparity calculations, it is very difficult to do any absolute measurements - the reason for this is that the origin (0,0,0) of the camera is unknown and can vary from camera to camera." [31]
4.3.2 Mathematical Framework
Chapter 5

Conclusion
Bibliography


