A Kinematic Self Calibration Model for Tool Use and Body Schema Learning

Mary Farner

Advisor: Brain Scassellati
Supervisor: Alessandro Roncone

Senior Project for B.S. in Computer Science & Mathematics
Yale University
May, 2016

Abstract

As the potential use of robots in everyday life expands from fantasy to reality, human-robot interaction (HRI) is at the forefront of robotics research. In order to create a robot that interacts safely and effectively with humans in existing environments, it is essential that the robot understands both its environment and itself. This perception of self should be developed and adjusted in the same way that the human perception of self develops throughout youth and adjusts over a lifetime. Implementing a robot system that emulates human systems as closely as possible allows for smooth cooperation between robot and human, as well as the robot’s autonomous interaction with and adaptation to the environment.

This project works with Baxter Research Robot to write a kinematic model for self calibration, allowing for self-correction and adaptation by Baxter in real life situations. A simulation was created in MatLab that models the calibration of the kinematic parameters of Baxter’s manipulator chains, based upon the measured position of end-effectors. The model uses the Denavit-Hartenberg parameters as a convention to represent end-effector position and orientation based upon link and joint dimensions.

The simulation was run on both Baxter’s real parameters alone and with the addition of an extra link representing a simple tool, and led to close convergence in both cases. While just a simulation, the next steps to implementation in the Robot follow closely.

I. Introduction

A consistent challenge in the field of robotics is emulating human intelligence and behavior in the way that robots interact with their environment and other players in that environment. Established cooperation between humans and robots is applicable in areas from manufacturing, to healthcare, to military operations. Thus, the field of human-robot interaction (HRI) has been widely studied in recent years with the goal of establishing safe, effective cooperation. Both a robot’s interaction with human peers and its self sufficient adaptability are factors in its effectiveness. In creating a human-like robot that operates with human-like behaviors and abilities, it is natural to study how humans learn and adapt to their environment, and to try to emulate that in the robot. Humans learn about their own bodies and the way they interact with the environment through experience [Rochat [1998]], establishing a model of what Hart and Scassellati refer to as “Ecological Self” [Hart and Scassellati 2011]. Another aspect of humanness is the ability to adjust
and adapt with slight changes in both environment and body. For example, as a person’s body slowly grows over time, they will make slight adjustments to their internal model of perception of self in order to continue to exist within an environment. Unlike a human, most robots are either hard-coded with all of the information they need, or learn with the aid of an instructor, limiting the adaptability and independence of the robot. Baxter Robot is an example of such a robot, as it learns to move within its environment via human intervention. This limits Baxter’s ability to exist autonomously in a changing environment, as well as make slight adaptations due to inaccuracies in sensors, camera systems, or parameter measurements. This project works to create a simulated model of kinematic self-calibration for Baxter to correct these slight inaccuracies, and begins to explore the possibility of adaptation for tool use in Baxter.

II. Problem

Given the actual position of its end-effectors, and the nominal kinematic parameters of its two manipulators, the goal is to write a model in which Baxter can calibrate to correct its parameter values. Further, Baxter should also be able to adjust and calibrate to the addition of a tool to the end of its manipulator.

The following sections will outline the conventions used for representation and calibration of Baxter’s manipulators. Background on the Denavit-Hartenberg parameters, as well as Baxter’s specific parameters and the system assumptions for this project will lay the groundwork for understanding how the model goes about calibrating the end-effector positions. Next, an explanation of the process of calibration through iteration will be presented, before presentation of the results of encoding these processes in various different models, from single link to Baxter’s full representation with a tool.

III. Denavit-Hartenberg Parameters

The Denavit-Hartenberg parameters are a convention for attaching reference frames to the links of a kinematic chain, namely a robot manipulator. The DH parameters allow for a standardized representation of each part in the manipulator chain, allowing for specific calculation of the orientation and position of the end-effector. A given chain is made up of links connected by joints, oriented in standard 3-dimensional Cartesian planes. For a given joint \( i \), the \( z \)-axis runs along the joint axis, the \( x \)-axis is along the common normal between \( z_i \) and \( z_{i-1} \) (the \( z \)-axis of the previous joint) and the \( y \)-axis follows from those two. There are four parameters for each joint in the chain: the link length \( a_i \), the link twist \( a_i \), the joint length \( d_i \), and the joint angle \( \theta_i \).

\[
\begin{align*}
    a_i &= \text{distance from } z_{i-1} \text{ to } z_i \text{ along } x_i \\
    \alpha_i &= \text{angle from } z_{i-1} \text{ and } z_i \text{ about } x_i \\
    d_i &= \text{distance from } x_{i-1} \text{ and } x_i \text{ along } z_{i-1} \\
    \theta_i &= \text{angle from } x_{i-1} \text{ and } x_i \text{ about } z_{i-1}
\end{align*}
\]

As referenced by the Handbook of Robotics [Siciliano and Khatib 2007], there are various forms of the convention that vary in the way that joints and links are referenced to each other. This project follows the standard convention: joint \( i \) is the joint between links \( i - 1 \) and \( i \). The \( z_{i-1} \) axis is along the axis of joint \( i \), and the \( x_{i-1} \) axis is along the common normal between \( z_{i-1} \) and \( z_i \). See Figure 1 for an example configuration [Legnani et al. 1996].

With reference to Figure [1] to compute the transform matrix from joint \( i - 1 \) to joint \( i \), one executes a translation of length \( d_i \) along the \( z_{i-1} \)-axis, a rotation through \( \theta_i \) about the \( z_{i-1} \)-axis, a translation of length \( a_i \) along the \( x_i \)-axis, and a rotation through \( a_{n} \) about the \( x_n \)-axis:

\[
\text{Trans}_{z_{i-1}}(d_i)\text{Rot}_{z_{i-1}}(\theta_i)\text{Trans}_{x_i}(a_i)\text{Rot}_{x_i}(a_{n}),
\]
where:

\[
\text{Trans}_{z_{i-1}}(d_i) = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & d_i \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\text{Rot}_{z_{i-1}}(\theta_i) = \begin{bmatrix}
\cos \theta_i & -\sin \theta_i & 0 & 0 \\
\sin \theta_i & \cos \theta_i & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\text{Trans}_{x_i}(a_i) = \begin{bmatrix}
1 & 0 & 0 & a_i \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\text{Rot}_{x_i}(\alpha_i) = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \cos \alpha_i & -\sin \alpha_i & 0 \\
0 & \sin \alpha_i & \cos \alpha_i & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

This series of multiplications is summarized by:

\[
i^{-1}T_i = \begin{bmatrix}
\cos \theta_i & -\sin \theta_i \cos \alpha_i & \sin \theta_i \sin \alpha_i & a_i \cos \theta_i \\
\sin \theta_i & \cos \theta_i \cos \alpha_i & -\cos \theta_i \sin \alpha_i & a_i \sin \theta_i \\
0 & \sin \alpha_i & \cos \alpha_i & d_i \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

The resulting rotation matrix, \( R_i \), is the first three rows and columns of \( T_i \):

\[
R_i = \begin{bmatrix}
\cos \theta_i & -\sin \theta_i & \sin \theta_i \sin \alpha_i \\
\sin \theta_i & \cos \theta_i \cos \alpha_i & -\cos \theta_i \sin \alpha_i \\
0 & \sin \alpha_i & \cos \alpha_i
\end{bmatrix}
\]

And the position vector, \( P_i \), is the first three rows of the last column of \( T_i \):

\[
P_i = \begin{bmatrix}
a \cos \theta_i \\
a \sin \theta_i \\
d
\end{bmatrix}
\]

Given the roto-translation matrix of each joint, \( i^{-1}T_i \), in order to get the orientation and position of the end effector of the entire chain, the individual joint matrices must be merged. To relate each of the \( i^{-1}T_i \) to each other and connect the chain, multiply \( i^{-1}T_i \) for \( i \) from 1 up to \( N \), where \( N \) is the number of joints in the chain:

\[
T_N = \prod_{i=1}^{N} T_{i-1}T_i
\]
IV. CALIBRATION

Calibration methods were drawn from The Handbook of Robotics, Chapter 14, Model Identification [Hollerbach et al., 2008]. We define \( \phi \) as the parameter vector, which holds the individual DH parameters of each joint. Note, that rather than \( \theta \) parameters, the model is given an offset parameter, \( a_i \). The offset is the built-in part of the joint angle that specifies the positioning of the encoders with respect to the system [Roncon et al., 2014]. \( \theta_i \) is the free variable for any given configuration of the manipulator, while \( a_i, d_i, \) and \( o_i \) are fixed values of the manipulator, and subject to calibration. For the purposes of calibration, \( \phi \) is an \( N \times 4 \) matrix.

\[
\phi = \begin{bmatrix} a & \alpha & d & o \end{bmatrix}
\]

Each column of \( \phi \) is a parameter vector containing the value of that parameter in each joint/link in the chain, from 1 to \( N \). The model is run on an initial set of guessed DH parameters over \( L \) measurements (indexed by \( l \)) of values for \( \theta_i, \ldots, \theta_L \). The end effector position \( y^l \) for a single set of \( \theta^l \) values is:

\[
y^l = f(\theta^l, \phi)
\]

This nonlinear function \( f \) is the forward kinematic function (cf. Equation [1]). Here, the variable \( \theta^l \) is vector of \( \theta \) values used in the given step. By considering \( \theta^l \) absorbed into the function \( f \), it becomes \( f^l(\phi) \). Then, the end-effector point for input values \( \theta^l \) is calculated as:

\[
y^l_c = f^l(\phi^l + \Delta \phi)
\]

Using the Taylor series expansion at \( \phi^k \) in order to closely estimate the slightly nonlinear function \( f^l \) at \( \phi^k \), we get:

\[
y^l_c = f^l(\phi^k) + \left. \frac{\partial f^l(\phi)}{\partial \phi} \right|_{\phi=\phi^k} \Delta \phi + \text{higher order terms} \ldots
\]

which simplifies to

\[
y^l_c \approx f^l(\phi^k) + A^l \Delta \phi
\]

Where \( A^l \) is the Jacobian, found by taking the partial derivative of \( f^l \) in terms of each parameter of \( \phi^k \):

\[
A = \begin{bmatrix} J_a & J_a & J_d & J_o \end{bmatrix}
\]

\[
J_{a_i} = x_i
\]

\[
J_{s_i} = x_i \times (p_N - p_i)
\]

\[
J_{d_i} = z_{i-1}
\]

\[
J_{b_i} = z_{i-1} \times (p_N - p_{i-1})
\]

The Jacobians are all based off of the roto-translational matrix from step \( i \) or \( i - 1 \) in Equation [1] or the parameters of the combined joints up to joint \( i \). Note that \( x_1 \) and \( z_1 \) are columns one and three of \( R_i \), representing the orientation about those axes. \( J_{d_i} \) is based on the \( z_{i-1} \)-axis because \( d \) runs directly along it, and the same is true for \( J_{b_i} \), and \( J_{s_i} \) is a function of \( x_i \) because it is oriented about this axis, and the same is true for \( J_d \) and the \( z_{i-1} \)-axis.

Assuming that the computed output, \( y^l_c \) is equal to the end effector on input \( l \), we can define the next step of the model in terms of the change in \( y^l \):

\[
\Delta y^l = A^l \Delta \phi
\]

where:

\[
\Delta y^l = y^l - f^l(\phi^k)
\]

Combining each of the \( \Delta y^l \) and \( A^l \) matrices for \( l = 1 \ldots L \) into two single stacked matrices, \( \Delta y \) and \( A \) respectively, we get the equation:

\[
\Delta y = A \Delta \phi
\]

From this, we can solve for an estimate of \( \Delta \phi \). If \( A \) were the square, full rank matrix of a linear model, the solution would be:

\[
\Delta \phi = A^{-1} \Delta y
\]

But \( A \) is not square, as this is a nonlinear model. To resolve this, we use the Moore-Penrose pseudo-inverse [Penrose, 1955]:

\[
A^\# = (A^TA)^{-1}A^T
\]
which applied to both sides gives:
\[(A^T A)^{-1} A^T \Delta y = (A^T A)^{-1} A^T A \Delta \phi\]
which simplifies to:
\[\Delta \phi = (A^T A)^{-1} A^T \Delta Y\]

Starting with the original input parameters, a new adjusted set of parameters is produced. \(\phi_{k+1}\) is defined on each iteration as
\[\phi_{k+1} = \phi_k + \Delta \phi\]
This process is repeated until \(\Delta \phi\) is sufficiently small (discussed later).

\[\begin{array}{cccc}
a & a & d & \text{offset} \\
0.069 & -\pi/2 & 0.27035 & 0 \\
0 & \pi/2 & 0 & \pi/2 \\
0.069 & \pi/2 & 0.36435 & 0 \\
0 & \pi/2 & 0 & 0 \\
0.01 & -\pi/2 & 0.37429 & 0 \\
0 & \pi/2 & 0 & 0 \\
0 & 0 & 0.229525 & 0 \\
\end{array}\]

Table 1: DH Parameters of Baxter’s arms

Figure 2: Baxter Robot in the Social Robotics lab at Yale University

V. Methods

In this project, a simulation of calibration on Baxter was implemented in MatLab. For the simulation, the estimated initial parameters as well as the theoretical actual parameters are given as input to the model, which then calibrates the initial guess parameters to converge toward the actual parameters based on the calculated end-effector position.

In reality, the actual end-effector positions would be measured, either by external sensors or Baxter’s own vision system, and given as input to the model. However, in the simulation, the actual position is calculated based on given parameters. That is, the simulation arrives at the actual end-effector position by calculations based upon the assumed parameters provided as \(\phi_{\text{actual}}\), whereas the calibration model would, in reality, be provided with the actual end-effector position, \(P_{\text{actual}}\). This does not change the way that calibration and convergence of the estimated parameters work, though, as the actual parameters are used only to calculate the actual end-effector position.

I. Baxter’s DH Parameters

Estimates of the Denavit-Hartenberg parameters for Baxter were taken from their nominal values \[1\]. Baxter has two chains (arms), each consisting of 7 joints. The two arms have identical parameters, representing the “shoulder”, “elbow”, and “wrist” joints. The only difference between the two arms is in the rotation matrix that specifies how it attaches to the torso, but for the purposes of simulation of end-effectors, this is not important. Thus, for simplicity, the calibration models was run on one arm only, as implementation in the second arm is identical. The Denavit-Hartenberg parameters for Baxter’s arms are listed in Table 1. See Figure 2 for a photo of the Baxter robot, and compare to Figure 3, two views of a graphical representation of Baxter’s arms. Note, there are 7 joints but three of them provide rotation only, and therefore do not have visible links.

II. Input Parameters

The inputs to the model are two parameter vectors representing the actual and initial guess DH parameters, the number of samples per it-

\[\text{https://groups.google.com/a/rethinkrobotics.com/forum/#!topic/brr-users/5Xi6wJa1I} \]
eration, and a set of lower and upper limits for parameters. The value for number of samples determines how many different configurations of $\theta$ values are to be tested in each step of calibration. Having enough samples is essential for the model to converge because there must be enough data for patterns to arise. The number of samples must grow for larger chains, so it is important to make this a flexible input to the model.

The upper and lower limits on parameter values are put in place to ensure physically realistic values (non-negative lengths, physically admissible joint angles) as well as to encourage convergence. For this purpose, the limits restrict parameters to within small percents of the actual value. Because this simulation is of calibration to correct small inaccuracies in parameters, it is safe to assume that such limits would be known and could be applied in a real world situation.

III. Reference Frame

The reference frame for calibration is rooted at $O_{-1}$, the origin of the camera frame. The first joint in the calibration calculation, joint 0, is not an actual joint but is rather an imagined connection between the origin of the camera system and the origin of the actual manipulator. For the purposes of simulation, it is assumed that for the actual parameters, $a_0$, $a_0$, and $a_0 + \theta_0$ are all set to 0, and $d_0$ is .1, or 10 centimeters. This means that coordinate system -1 and 0 are oriented identically, with a .1 meter shift along the $z$ axis. The only difference for the guess parameters is that $d_0$ is set to 0 as well. From $O_0$ onward, the coordinate systems are based upon the parameters of the links and joints in the robot manipulator. Refer back to Figure 1 for a visual understanding of the reference frames.

IV. Convergence Limits

In order to properly converge, the initial guess for the parameters must be relatively close to the actual values, as the model simulates small inaccuracy corrections. Iteration stops when the difference between $y$ to the actual position becomes minimal. Convergence is considered complete when the norm, or magnitude, of $\Delta y$ reaches 0. For the simulation, an exact zero value is never reached, so the limit of convergence is set to $10^{-30}$.

V. Simulating Tool Use

The use of a tool, once the robot is accurately and fully calibrated, is analogous to simply
adding another link to the chain. For the simulation, we assume for simplicity without loss of generality that in Baxter, the last link of either arm is positioned with the z-axis running outward, parallel to the axis of the end-effector. Thus, the addition of a simple tool can be simulated by adding another link, assume of length 10 cm, with the only parameter of interest being the link length, \( a \). All other parameters are set to zero. This represents the tool of length 10 cm extending directly out of the robot’s end-effector, thus having a joint angle and twist of 0° and a joint length of 0 cm.

VI. EXPERIMENTAL RESULTS

I. Single Link Convergence

With relatively small chains of one to two links, convergence happened quickly and with a relatively low number of samples and iterations. Figure 4 shows the output for a simple one-joint model, with convergence after just seven iterations. Note that parameters indexed by 0 are the imaginary joint for establishing orientation. We see convergence in \( d_0 \) and \( o_1 \) toward the correct parameters, as the output values are close to the actual despite differing guess values. Establishment of this simple model’s success allowed for building up to the full representation of Baxter’s 7 links.

II. Convergence With Baxter

The extension to Baxter’s manipulator required more sample measurements and more iterations, but resulted in equally accurate convergence. Figure 5 depicts the output of running calibration over 1,000 samples and 47 iterations, on parameters that differ (arbitrarily) from the actual in \( d_0, o_1, o_4, \) and \( o_5 \). These rows are highlighted in the figure, and the convergence is reflected in the change from guess to output.

![Figure 4: Output of the model run on just one joint. Clear convergence in the two inaccurate guess parameters](Image)

![Figure 5: Output of the model run on 1,000 samples given Baxter’s complete parameter set](Image)
run identically on both the actual and output parameters. Clearly, the two end-effector positions are the same, confirming convergence and accurate calibration.

III. Simulation of Tool Use

For the addition of a tool, the guess and actual parameters were both identical to Baxter’s actual parameters, with the addition of an 8th joint that varied in one parameter. The actual parameters were set to

\[
a_i = .1\quad a_i = 0\quad d_i = 0\quad \theta_i = 0
\]

and the guess parameters were all set to 0. This simulated a 10 cm long tool extending directly out of Baxter’s end effector. Convergence for the tool was achieved very quickly, again using 1,000 samples, as the only adjustment to make was in the \(a\) parameter. See Figure 8 for the output of the model, noting the single highlighted row as the only changing parameter.

Figure 6: A graphical representation depicting both the actual and calibrated manipulators. The arm resulting from actual is gray, the calibrated blue, and clearly they are nearly the same. Generated with code adapted from Alessandro Roncone. [Roncone, 2013]

Figure 7: Further evidence of the convergence of the actual and calibrated parameter was found by calculating their identical end-effector positions via the model.

Figure 8: The output of the model run on 1,000 samples given Baxter’s complete parameter set, as well as the addition of one joint representing a tool. Convergence occurred very quickly here.
Figure 9 is a graphical representation of the two arms. Note that because the chains are identical up to the tool, the two arms overlap and appear as one in the representation. Finally, Figure 10 is the result of running the forward kinematics script on the actual and calibrated parameters of the manipulator with a tool, confirming that the end-effector calculation are identical and calibration was successful.

Figure 9: The arm with drawn from the actual parameters is gray, the output parameters were used to draw the blue arm; however, they are so close that they appear as one. Generated with code adapted from Alessandro Roncone. [Roncone, 2013]

```
>> DH_Model(actualB, theta_values, ulimitsB, llimitsB);

end_effector =

  0.3924
-0.4141
0.2361

>> DH_Model(phi, theta_values, ulimitsB, llimitsB);

end_effector =

  0.3924
-0.4141
0.2361
```

Figure 10: Evidence of identical end-effector positions with the addition of a tool.

VII. Further Work

The project established a working simulation of kinematic self-calibration in the Baxter robot, setting the foundation for next steps. First, the model can be moved out of MatLab, into the Robot Operating System (ROS) to be implemented on Baxter. Initially, the position of the end-effectors will be detected by an external camera system or some other form of report, so that the input to the model mirrors the reality of Baxter detecting the position of his own arm. Once a working model of calibration is established in Baxter, the final step will be to connect Baxter’s own vision system to detect the actual position of the end-effector to use as a model for self-calibration. Execution of this step would result in fully autonomous and automatic self-calibration capabilities for Baxter.

Outside of calibration, further work will also include incorporating the entire process of human-like learning in Baxter. That is, starting with no information, writing a model that allows Baxter to establish a model of his own body, his environment, and the way the two interact. Then, once Baxter’s overall kinematics have been approximated, the self-calibration model will run automatically and continuously to continue adapting and perfecting. A method that follows the work done by Hart and Scassellati in another robot would be closely followed [Hart and Scassellati, 2011]. They used Circle Point Analysis (CPA) to step through each joint in the manipulator chain, establishing each joint’s parameters one at a time and in order. The process starts by establishing a certain “home position” for the manipulator that is easily and accurately reached. Then, for each joint, the manipulator is placed in this home position, and the joint in question is assessed using CPA in order to establish its parameters. Once a joint is finished, the process moves to the next joint, having established an accurate model of all previous joints from which to build [Hart and Scassellati, 2011]. For Baxter, a similar process can be used, using this calibration model rather than CPA in each joint step, allowing Baxter to build a working
kinematic model from scratch.

VIII. CONCLUSIONS

While this project only produced a simulation of the kinematic self-calibration model, it is a step toward full autonomous learning and an example of the possibilities in this area. As robots become more essential to human life, and thus begin to interact more and more with humans, perfecting the art of HRI is essential. It is important that robots think and act as much like humans as possible if they are to be integrated into human-run processes, and adaptability and autonomy will be vital. A robot which depends upon human input and support has a very narrow scope limited to controlled, niche environments, so creating models that allow robots to adapt on their own will lead to new opportunities for collaboration between robots and humans.

Acknowledgments

Special thanks to Alessandro Roncone for all of the support and patience throughout the semester, providing helpful code and testing practices, and pushing me to arrive at answers on my own.

Code

The code repository can be found [https://github.com/ScazLab/baxter_self_modeling.git](https://github.com/ScazLab/baxter_self_modeling.git)

REFERENCES


Alessandro Roncone. Kinematics visualization on matlab, 2013. [Source Code on GitHub)]

