2018-07-01

Tactile Sensing and Position Estimation Methods for Increased Proprioception of Soft-Robotic Platforms

Nathan McClain Day
Brigham Young University

Tactile Sensing and Position Estimation Methods for Increased Proprioception of Soft-Robotic Platforms

Nathan McClain Day

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Marc Killpack, Chair
Tim McLain
Steven K. Charles

Department of Mechanical Engineering

Brigham Young University

ABSTRACT

Tactile Sensing and Position Estimation Methods for Increased Proprioception of Soft-Robotic Platforms

Nathan McClain Day
Department of Mechanical Engineering, BYU
Master of Science

Soft robots have the potential to transform the way robots interact with their environment. This is due to their low inertia and inherent ability to more safely interact with the world without damaging themselves or the people around them. However, existing sensing for soft robots has at least partially limited their ability to control interactions with their environment. Tactile sensors could enable soft robots to sense interaction, but most tactile sensors are made from rigid substrates and are not well suited to applications for soft robots that can deform. In addition, the benefit of being able to cheaply manufacture soft robots may be lost if the tactile sensors that cover them are expensive and their resolution does not scale well for manufacturability. Soft robots not only need to know their interaction forces due to contact with their environment, they also need to know where they are in Cartesian space. Because soft robots lack a rigid structure, traditional methods of joint estimation found in rigid robots cannot be employed on soft robotic platforms. This requires a different approach to soft robot pose estimation. This thesis will discuss both tactile force sensing and pose estimation methods for soft-robots. A method to make affordable, high-resolution, tactile sensor arrays (manufactured in rows and columns) that can be used for sensorizing soft robots and other soft bodies isReserved developed. However, the construction results in a sensor array that exhibits significant amounts of cross-talk when two taxels in the same row are compressed. Using the same fabric-based tactile sensor array construction design, two different methods for cross-talk compensation are presented. The first uses a mathematical model to calculate a change in resistance of each taxel directly. The second method introduces additional simple circuit components that enable us to isolate each taxel electrically and relate voltage to force directly. This thesis also discusses various approaches in soft robot pose estimation along with a method for characterizing sensors using machine learning. Particular emphasis is placed on the effectiveness of parameter-based learning versus parameter-free learning, in order to determine which method of machine learning is more appropriate and accurate for soft robot pose estimation. Various machine learning architectures, such as recursive neural networks and convolutional neural networks, are also tested to demonstrate the most effective architecture to use for characterizing soft-robot sensors.

Keywords: tactile sensing, machine learning, cross-talk compensation, parameter-free learning, parameter-based learning, soft robot sensing, convolutional neural networks, recursive neural networks

ACKNOWLEDGMENTS

I would like to thank Marc Killpack for all of his guidance and direction throughout all aspects of my research, for his patience and help, and for giving me the opportunity to work in such a wonderful lab. I really grew because of his excellent tutelage. I would also like to thank each of my lab mates for always being willing to help come up with solutions and answers to problem I would come across. They were more than just lab mates, each of them was an excellent friend and mentor. Last of all, and most importantly, I want to thank my wonderful wife for always supporting me, and helping me believe that I could actually finish this thesis. She always encouraged me to work for things I never thought I would be able to accomplish. She rooted for me and knew that I could be more than what I believed I could be. I will always be grateful to her for her patience and support through all of this.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

1.1 Need for Soft Robots and Soft Robot State Estimation

Because of their accuracy and strength, industrial robots have been used for decades. Their main drawback is their inability to interact safely with human beings or to work in fragile environments. Because of their high inertia and strength, industrial robots are often isolated from humans for safety purposes. This isolation inhibits robots from assisting humans in daily tasks, or working in unmodeled environments where unintended contact is likely to occur which could cause damage to the robot or its environment. As an example, traditional robots could be very useful in space, but are heavy and likely to damage sensitive equipment. Additionally, manufacturing industries could benefit from automating processes that cannot be done with industrial robots, and must be performed by hand, such as sanding airplane hulls for painting, or processing foods. A final example of where traditional robots cannot be used is in assisted living, where some adults no longer have the ability to perform their daily tasks.

The limitations discussed previously have given rise to a new field of research, namely soft robotics. This subfield of robotics encompasses robots made from soft, deformable materials such as plastic, fabric, or compliant mechanisms. Soft robots are beneficial because they have very low inertia, allowing them to interact and make contact with their environment without causing damage to themselves, humans, or their surroundings.

1.2 Problem Description

Even though soft robots have the potential to overcome some of the drawbacks of rigid industrial robots, they have their own limitations. To effectively interact with their surroundings, they need to be able to sense objects around them. One benefit of soft robots is that they can seek out and make contact with their environment without damaging themselves or objects they

interact with. There are two main methods of sensing that would allow soft robots to work in the environments described above. The first would be the ability to seek out and make contact with objects through tactile sensing, and the second would be deflection and bending estimation through improved joint or link orientation sensors. Because soft robots inherently lack rigidity, predicting and measuring their respective link orientations is a difficult task. Due to this lack of rigidity, traditional link orientation estimation methods typical for rigid robots cannot be implemented for soft robots. This requires new methods of estimating soft robot joint configuration and link orientation that have not been developed.

1.3 Related work

1.3.1 Tactile Sensing

Related to my work in this thesis, other researchers have begun to examine different tactile sensing methods for soft bodies. Rogers et al. [1] discuss different types of materials that are used in stretchable electronics, and mainly focus on organic-based devices such as those discussed in [2]. A discussion of fabric-based tactile sensors was not included. Other examples of flexible sensors are shown in Someya et al. [3] and [4] demonstrating a conformable, flexible network of pressure sensors using organic transistors. The manufacturing of such sensors typically requires a clean room and sophisticated prototyping processes and equipment along with domain-specific expertise. The use of transistors also adds complexity to the circuit.

Further examples of tactile sensors include Yamaguchi et al. [5], who have shown that they could use optical sensing of a transparent soft material on a gripper to improve performance while gripping tools. Sareh et al. [6] have used tactile sensing on a soft robot prototype for minimally invasive surgeries. Although both of these tactile sensor designs were created specifically for soft robots, it is unclear if they would generalize and scale well for large surface areas and soft bodies, as we show in this paper.

When resistance based sensors are constructed using a grid of rows and columns, cross-talk can occur which can throw off sensor readings. Cross-talk was mentioned in [3], but a solution for cross-talk compensation was never proposed.

Specifically related to the variable resistance tactile sensing method that we present in this thesis, many researchers have explored methods of tactile sensing using voltage drive lines and analog to digital conversion lines set in a grid to decrease the number of electronics and increase resolution. The difficulty with this architecture is that the sensing locations, called tactile pixels or taxels, are no longer isolated circuits and cross-talk occurs. Two types of cross-talk are possible: mechanical cross-talk and parallel parasitic cross-talk. Mechanical cross-talk is caused by indirectly straining material when pressing on an adjacent taxel. This effect was discussed in Canavese et al. [7]. However, Canavese et al. never tested for parallel parasitic cross-talk. Parallel parasitic cross-talk is caused by resistors in parallel that influence the voltage output when two or more non-adjacent nodes are compressed simultaneously. An example of when this type of cross-talk would occur is shown in chapter 2 Figure 2.1.

Past methods for cross-talk compensation, although effective at eliminating current leakage (see [8, 9]), do not eliminate parallel parasitic cross-talk. Other research that has attempted to address this problem requires complicated circuitry which increases cost and complexity for manufacturing (see [10–12]).

Our initial tactile sensors were based on work performed by researchers at the Georgia Institute of Technology [13]. Fabric taxels were individually sewn and wired, which limits the scalability of the design due to its labor intensive nature and increased cost and circuit complexity. Another problem with this design is that when isolating each taxel electrically, the sensing resolution is very limited. This is due to the size of each taxel and due to the electronics required for a large array of sensors. The taxels can be made smaller, but then the number and complexity of electronics needed to get information from each taxel greatly increases. For example, in [13] since each taxel requires a single analog-to-digital converter (ADC) - 100 taxels would require 100 ADCs. However, our methods allow us to design and build a sensor that, in addition to being flexible, low-cost, and easily made in-house, also scales well to large areas and does not require complex circuitry.

1.3.2 Pose Estimation

Research into embedded sensors for soft-robot position estimation has been increasing as many labs realize the need for closed loop control of soft-robots. Typically position feedback for

soft robots is done using motion capture systems. Some motion capture systems employ infrared cameras and reflective tracking dots, such as those used in [14], while others may use electromagnetic field detectors such as the one used in [15]. The issue with using this type of system is that it constrains the mobility of the soft-robot to a confined area limited by the motion capture system setup. This severely reduces the mobility of any platform that uses soft-robots with feedback control capabilities. Since soft robots are meant to be mobile and interact with their environment, their structures must be able to make contact with their environment without damaging themselves or their surroundings. This means that soft-robot structures inherently lack rigidity, which makes it difficult to attach sensors, such as rotary encoders, to traditionally rigid connections such as pin joints. This requires the development of soft sensing techniques. Some methods that have been developed and implemented on soft robots include flex and bend sensors [16] [17] [18] [19], photodiode sensors [20], inductance sensors [21], and fiberoptics [22].

Elegeneidy et al. embedded a flex sensor into a silicon based soft robotic finger [18]. One issue that stretch and flex sensors have is mechanical wear, mechanical creep, and voltage drift. To avoid the effects of voltage drift, Elegeneidy et al. decreased the actuation time for the soft robotic finger to 0.5 seconds. This was not enough time to demonstrate the effects of voltage drift. If the flex sensor is held for a period of time at a constant bend angle, the measured voltage drifts as the strain in the sensor relaxes due to mechanical creep in the material. This material behavior makes this type of sensor unsuitable for extended use. In [16], Yuen et al. used a capacitive stretch sensor instead of the traditional resistance based sensors to capture a bend angle. Using this type of sensor they were able to capture bend angle as well as displacement due to buckling in the passive linkages. Though this sensor is effective in estimating rotation about a single axis, it would not be able to estimate motion outside of a two dimensional plane, which can occur due to unconstrained motion typical of soft-robots.

Felt et al. employed inductance from current passing through coiled wires to detect changes in bend angle [21]. A voltage would be applied to coiled wires wrapped around a soft robot joint. As the distance between coils increased or decreased the inductance would increase or decrease. This allowed them to correlate inductance with joint angle. Although well suited for soft-robot applications, this method only allowed for sensing about a single axis and can also be affected by

induced magnetic fields from nearby wires as well as ferrous or magnetic materials that the robot may come in contact with.

One method that is not affected by proximity of magnetic fields or mechanical creep uses photo-diodes and LED's. Yuan et al. were able to measure bend angle using fiber optic cables and measured light intensity [22]. Here they had 4 fiber optic cables strung together. As the cable was bent or twisted four distinct light patterns would impinge on a sensing surface. The shape of the four light patterns could be detected and correlated directly to bend and twist angles. Though this development of this sensor was not for soft-robotics specifically, this type of sensor could definitely be used as a viable solution for position estimation. Similar work was done in [20] where a photo-emitter and photo-diode were used for sensing, but the emitter and diode were not joined by a fiber-optic cable. As the bend angle changed, the measured intensity from the photo-diode would also change. One flaw of this method is that it is difficult to differentiate between positive and negative angles, because the intensity at -30° can measure the same as the intensity at +30°. This results in the sensor only being useful for measuring angles between 0° and 90° about a single axis of rotation.

You et al. applied machine learning to a soft-robot manipulator [23]. Their approach differed from approaches previously discussed, in that they were not using a position feedback sensor. Their "feedback" was composed of the outputs from a machine learning algorithm which were calculated from the robots actuation pressure readings. Their neural network would take in actuation pressures and output estimated position in a 2 dimensional plane. This method would not work well for unmodeled disturbances, because there may not be enough information in pressure alone to account for unmodeled disturbances. This would mean that a controller would have inadequate feedback or be responding to an incorrect estimate.

Melingui et al. also took a different approach to sensing. Instead of coming up with a traditional sensor calibration scheme, they employed machine learning to estimate position based on sensor outputs [24] [25]. They fastened string potentiometers along the length of a Compact Bionic Handling Assistant's (SBHA) trunk to obtain position feedback. They then used machine learning using a fully-connected layer to come up with a calibration due to the difficulty of developing an accurate analytical model for the CBHA. Instead of developing a calibration for their sensors, Melingui et al. used the sensor feedback information to directly produce inverse kinematics. Their

approach differed from ours in that their machine learning method used hidden layers that took in the current input instead of looking at a time series sequence.

1.4 Thesis Overview

Chapter 2 of this thesis focuses on a tactile sensor that increases the resolution of fabric-based tactile sensors. Because of the grid design used of sensing rows and columns, the sensor suffers from crosstalk. Two methods for crosstalk compensation are discussed. The first method includes a model developed to predict crosstalk which is used to estimate resistance at each sensing location. This estimated resistance is then directly correlated to force. The second method uses diodes and a multiplexor to isolate sensing locations. The results from these two methods are compared.

Chapter 3 discusses various proposed mehtods of end-effector position estimation for soft-robot platforms. Specifically, we focused on the characterization of a string potentiometer sensor array using machine learning. Parameter-based versus parameter-free machine learning methods are compared for various Recursive Neural Network architectures are well as Convolution Neural Networks. convolutional neural networks outperform recursive neural networks. For our soft robot position estimation problem, we also determined that parameter-free methods are more accurate than parameter-based methods when disturbances are present.

Finally, in Chapter 4 we discuss the results from Chapter 2 and Chapter 3. Future work for each topic and how research can continue on in the future is also discussed.

1.5 Specific Contributions

Specific contributions to the soft-robotics field include two new methods of cross-talk compensation for resistance based tactile sensors. The first method being the model-based method developed in chapter 2 as well as the diode method discussed in the same chapter

Contributions relating to end-effector position estimation include determining that convolutional neural networks perform better than recursive neural networks when characterizing soft-robot motion. Parameter-free machine learning was also determined to be a more robust and reliable way of learning for increased estimation accuracy.

CHAPTER 2. TACTILE SENSING AND CROSS-TALK COMPENSATION

2.1 Tactile Sensing

Tactile sensing has been developed for decades because of its promise to improve a robot's ability to interact with the world using closed-loop feedback control. One of the most popular methods of tactile sensing uses material that changes resistance when compressed in series with a constant value resistor. This results in a voltage divider circuit that allows a changing voltage to be read from the node between the two resistive elements. This method is demonstrated in multiple papers [7, 11, 13, 26–28].

Our goal in the design and manufacturing of our sensor was to make it scalable as well as soft and deformable for integration with a soft robot. Integration on a human arm was also explored with the help of Jimmy Pena and Veronica Santos from UCLA [29]. The sensor presented in this thesis is made of conductive, non-conductive, and resistive fabric materials. The voltage output of each taxel (or individual tactile sensor unit) is directly correlated with how much the resistive fabric is compressed.

To decrease the amount of electronics required while increasing sensing resolution, we built a sensor that operates in a grid with rows and columns (see Figure 2.1). A single column is powered with 5 volts, while simultaneously reading the voltage along a single row using an analog-to-digital converter (ADC). Where the column and the row intersect, a taxel is formed, and a voltage reading can be obtained and correlated to force. This method increases the sensing resolution while decreasing the amount of electronics that would be required for individual sensors. However, this form of sensor array may exhibit crosstalk when multiple taxels are activated simultaneously. Addressing this problem is one contribution of our work. Two methods are introduced: the use of a mathematical model to calculate changes in resistance, and the minimal use of additional simple circuit components to electrically isolate taxels.

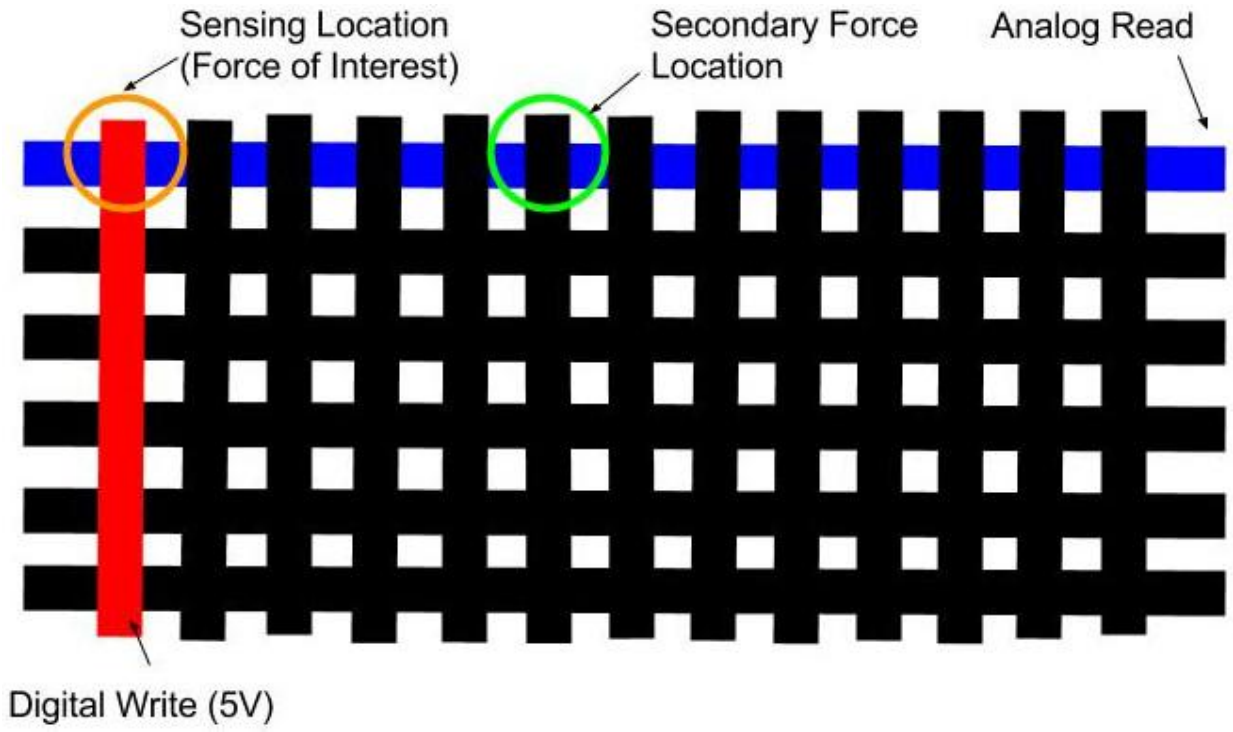


Figure 2.1: Representative scenario for parallel parasitic crosstalk.

2.2 Sensor Construction

Individual taxels were typically constructed following the design shown in Figure 2.2, similar to those described in [13]. We employed the same materials used in [13] in the new grid-style sensor.

We used an iron-on adhesive to glue the conductive fabric to the non-conductive spandex material. Strips of conductive fabric were glued to one side of the spandex material in rows, while columns of the conductive fabric were glued to a spandex fabric opposite. A large piece of resistive material (EeonTex LG-SLPA-20K supplied by Eeonyx) was then placed between the two layers of spandex with the columns and rows of conductive material facing the resistive material (see Figure 2.3).

Metal snaps were connected to the ends of the conductive fabric strips so that wires could be soldered to them. We used an Arduino Mega's ADC channels to record from the rows and we used the digital outputs to power the columns of the sensor. To obtain information from each taxel using the Arduino, we read from every ADC channel while cycling through each 5V digital I/O

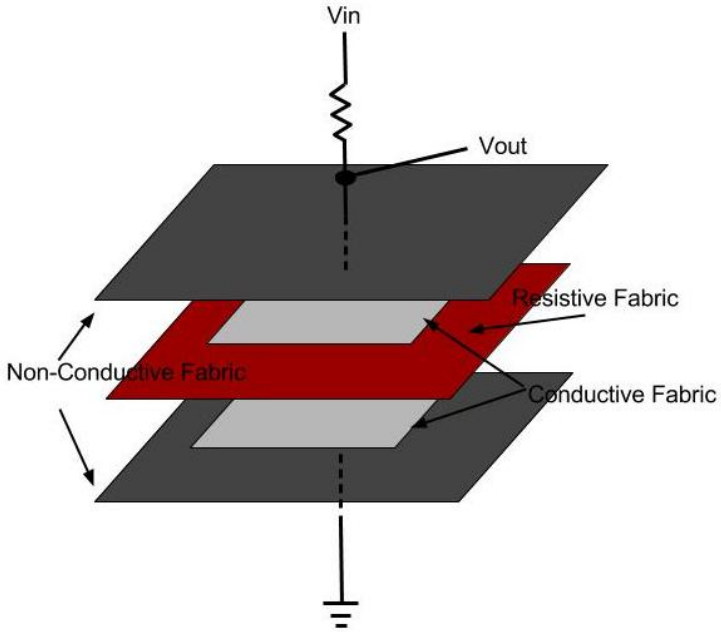


Figure 2.2: Construction of an individual taxel.

Figure 2.3: Sensor construction with columns and rows of conductive fabric. The gray resistive material is folded back to reveal columns of conductive material. Metal snaps are located at the top of each column.

pin. Using 27 I/O pins and 11 ADCs we were able to read from 297 taxel positions at a rate of approximately 80 Hz with un-optimized code on the Arduino Mega.

There are some limitations for our current hardware design. The minimum spatial resolution is limited by the conductive filament used. For the first application we show on a soft robot,

we used 3/8 in. strips of conductive fabric, but there is conductive thread that is available and should also work in a similar manner. Additionally, since we currently use metal snaps, our row and column sizes are limited by the size of the snaps. However, that is not a fundamental limitation and could be resolved if finer spatial resolution is needed. Finally, in terms of sampling real-world phenomenon at high frequencies, we fully expect the mechanical response of the fabric to be much slower than the electrical response. Therefore the limit to sampling any kind of impulsive force would likely be the actual fabric mechanics or the sampling rate of the ADC.

2.3 Modeling Cross-talk

Initial tests of a grid tactile sensor prototype showed that there was a significant amount of parallel parasitic crosstalk when compressing several taxels in the same row. To compensate for this crosstalk, we developed a model of the circuit that would estimate the behavior of the crosstalk. Figure 2.4 shows a simplified circuit schematic of a three row by three column grid sensor. R_{kj} for $k = 1,2,3$ and $j = 1,2,3$ represents the changing resistance of each node when the resistive fabric is compressed. R_{C_1} through R_{C_3} represent the constant value resistors needed to form a voltage divider. V_1, V_2, and V_3 represent the alternating square waves used to power each column individually and V_{out1}, V_{out2}, V_{out3} represent the location where the voltages are read by the ADCs.

Using Kirchoff's current and voltage laws and this simplified 3×3 version of the high resolution tactile sensor, our model showed that the voltage output, V_{out}, is a function of all the resistance values corresponding to the same row. For example, V_{out1} only depends on the resistance values of the first row. Equation 2.1 shows the form of this relationship.

$$V_{out1} = \frac{aV_3 + bV_2 + cV_1}{R_{11}R_{12}R_{13} + a + b + c} \tag{2.1}$$

Where:

$$a = R_{11}R_{12}R_{C_1}$$

$$b = R_{11}R_{13}R_{C_1}$$

$$c = R_{12}R_{13}R_{C_1}$$

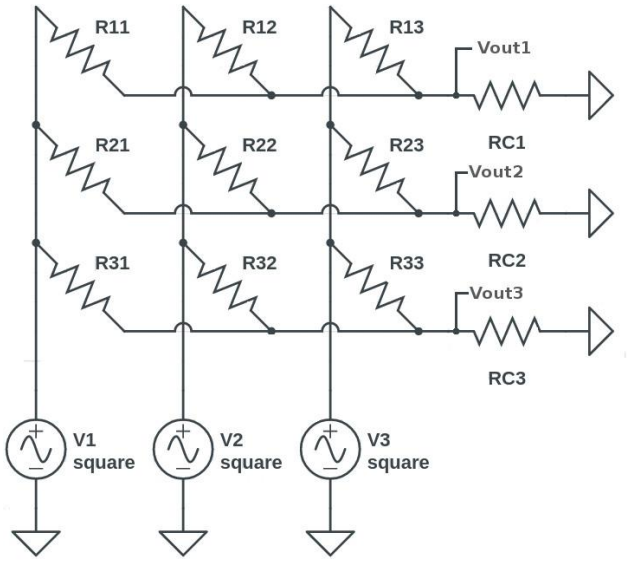


Figure 2.4: 3×3 Simplified High Resolution Sensor.

Since each voltage input is activated one at a time using alternating square waves, the Equation simplifies depending on which channel is powered. If $V_1 = 5$ V, then $V_2 = V_3 = 0$, Equation 2.1 then simplifies to:

$$V_{out1,1} = \frac{cV_1}{R_{11}R_{12}R_{13} + a + b + c} \tag{2.2}$$

Where $V_{out1,1}$ would represent the voltage reading at node R_{11} when only V_1 is powered. This shows that the voltage reading of one row is not only dependent on the change of resistance between where the column and row intersect, but on each change of resistance that occurs along the entire row. This is what makes it impossible to directly correlate force to V_{out}.

Cross-talk Simulation

To check the validity of Equation 2.2 and our ability to estimate crosstalk, we built a simple simulation in MATLAB to see if the crosstalk model behaved like the crosstalk in the physical sensor. The experiment consisted of holding two resistor values on a single row constant in simulation while increasing pressure on the first variable resistor on the same row. Without changing the force on the two resistors, the voltage readings dropped as force on the first resistor increased. This demonstrated the effect of parallel parasitic crosstalk. We then compared the results from our

simulation to data collected from a small-scale tactile sensor (3 rows by 3 columns) using the same method. This time, we applied a constant force or pressure to the first two taxels to get a constant resistance while simultaneously increasing pressure on the first resistor in the same row. Figure 2.5 shows that the simulated and actual responses were comparable. Any apparent differences were such that we expected to be able to compensate for them through calibration.

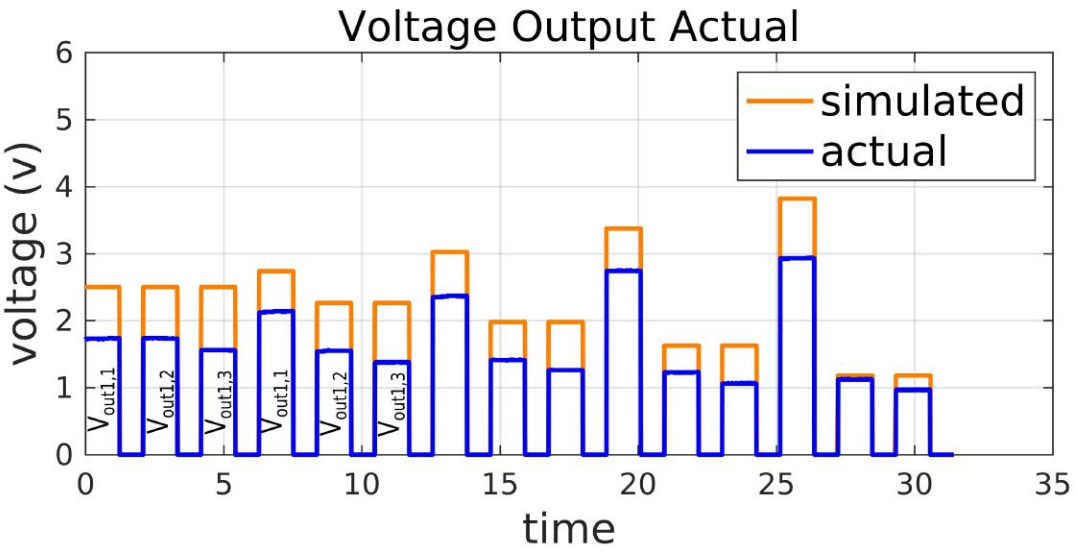


Figure 2.5: A comparison of simulated and actual tactile sensor crosstalk.

Based on the results of the above comparison, we saw that the voltage outputs were a function of all resistors corresponding to the same row. Since the actual resistance at each node is independent of all other resistances, we wanted a method to solve for the resistance directly based on the voltage readings obtained from each row.

2.4 Cross-talk Compensation

Based on the results of the above comparison, we saw that the voltage outputs were a function of all resistors corresponding to the same row. Since the actual resistance at each node is independent of all other resistances, we wanted a method to solve for the resistance directly based on the voltage readings obtained from each row.

From Equation 2.2 we can see that V_{out} is only a function of the row resistances and a known input voltage which allows us to analyze the circuit of a single row at a time. Figure 2.6 shows

what a single row circuit would look like with R_{kj} representing the j number of resistances of the k^{th} row, R_{C_k} representing the constant resistor in the voltage divider, V_j representing the cycling square waves, and $V_{out,kj}$ representing the voltage output on the k^{th} row when the j^{th} column is powered.

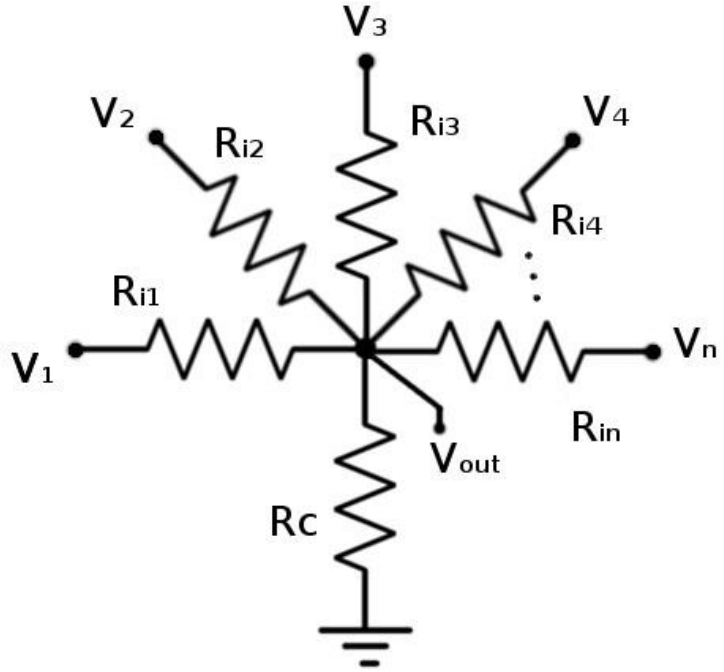


Figure 2.6: A simplified circuit showing a single row as a branching circuit.

Our known variables are V_k and $V_{out,kj}$ at time t (where t is a discrete time step), and R_{C_k} which is fixed. The goal is to find the R_{kj}'s. Given the voltage drop across a resistor ΔV, we know that $\Delta V = iR$, and that the sum of the currents into a node is equal to the sum of the currents leaving. Therefore, we can show that the current through the constant resistor, i_{C_k}, is

$$i_{C_k} = \frac{V_1 - V_{out,k}}{R_{k1}} + \frac{V_2 - V_{out,k}}{R_{k2}} \ldots + \frac{V_n - V_{out,k}}{R_{kn}} \tag{2.3}$$

At time $[t]$, V_1 is on and the rest of the input voltages are 0, and we sample $V_{out,k}[t]$. At time $[t+1]$, V_2 is on and the rest of the input voltages are off, and we sample $V_{out,k}[t+1]$, etc. A matrix, $\mathscr{V}_{\Delta,k}$, can be constructed for time $[t]\ldots[t+(n-1)]$ where n is the total number of columns. We can then create an $n \times n$ matrix of ΔV's for calculating the resistances of a single row as follows:

$$\mathscr{V}_{\Delta,k} = \begin{bmatrix} (V_1 - V_{out,k}[t]) & -V_{out,k}[t] & \dots & -V_{out,k}[t] \\ -V_{out,k}[t+1] & (V_2 - V_{out,k}[t+1]) & \dots & -V_{out,k}[t+1] \\ \vdots & \vdots & \ddots & \vdots \\ -V_{out,k}[t+(n-1)] & -V_{out,k}[t+(n-1)] & \dots & (V_n - V_{out,k}[t+(n-1)]) \end{bmatrix} \tag{2.4}$$

Using what we know from Equation 2.3, we get current if we divide ΔV by R. We define $\mathscr{R}_k$ as the array of inverse resistances.

$$\mathscr{R}_k = \begin{bmatrix} \frac{1}{R_{k1}} & \frac{1}{R_{k2}} & \frac{1}{R_{k3}} & \dots & \frac{1}{R_{kn}} \end{bmatrix}^T \tag{2.5}$$

The current i_{C_k} at any time $t = 1 \dots n$ is known and can be shown to be

$$\mathscr{I}_{C_k} = \begin{bmatrix} \frac{V_{out,k}[t]}{R_{C_k}} & \frac{V_{out,k}[t+1]}{R_{C_k}} & \frac{V_{out,k}[t+2]}{R_{C_k}} & \dots & \frac{V_{out,k}[t+(n-1)]}{R_{C_k}} \end{bmatrix}^T \tag{2.6}$$

Given these matrices, we can show that the following is true:

$$\mathscr{I}_{C_k} = \mathscr{V}_{\Delta,k}\mathscr{R}_k \tag{2.7}$$

Remembering that the unknowns we wish to find in this Equation are the inverse of the elements in $\mathscr{R}_k$, they can be solved for in the following manner:

$$\mathscr{R}_k = (\mathscr{V}_{\Delta,k})^{-1}\mathscr{I}_{C_k} \tag{2.8}$$

This solution only represents the inverse of the resistance values for a single row, k. All the resistances can be solved for simultaneously if we apply this method to each row. A block matrix can be formed using this solution for the k^{th} row. Using this matrix we can find the resistance values for every node. The form of this solution is shown in Equation 2.9 where m represents the total number of rows.

$$\begin{bmatrix} \mathscr{I}_{C1} \\ \mathscr{I}_{C2} \\ \mathscr{I}_{C3} \\ \vdots \\ \mathscr{I}_{Cm} \end{bmatrix} = \begin{bmatrix} \mathscr{V}_{\Delta,1} & \emptyset & \emptyset & \emptyset & \emptyset \\ \emptyset & \mathscr{V}_{\Delta,2} & \emptyset & \emptyset & \emptyset \\ \emptyset & \emptyset & \mathscr{V}_{\Delta,3} & \emptyset & \emptyset \\ \emptyset & \emptyset & \emptyset & \ddots & \emptyset \\ \emptyset & \emptyset & \emptyset & \emptyset & \mathscr{V}_{\Delta,m} \end{bmatrix} \begin{bmatrix} \mathscr{R}_1 \\ \mathscr{R}_2 \\ \mathscr{R}_3 \\ \vdots \\ \mathscr{R}_m \end{bmatrix} \tag{2.9}$$

2.5 Sensor Calibration

To illustrate the performance of the sensor after applying the result from Equation 2.9 to the raw data, Supplemental Video 1 in [29] shows us compressing multiple taxels in the same and different rows on a flat surface (https://youtu.be/_UjPh-Tb-cs).

The next step was to relate the measured values to force. Instead of using voltage in the crosstalk compensation code, we used the actual ADC readings reported from the Arduino Mega while assuming that the conversion between voltage and ADC values could be included in our calibration. For example, instead of using 5 V for V_k we substituted in the 10-bit ADC representation of 5 V, and instead of $V_{out_{k,j}}$ being reported as values between 0 V and 5 V, they were reported as values between 0 and 1024. One further note is that the sensor performs better when a low value resistor is used for the constant value resistor R_{C_k} since a lower resistor value increases the voltage range of each taxel. 330 Ω resistors worked well for our specific setup in the first demonstration application on a soft robot.

The tactile sensor was calibrated on a flat, rigid surface by pressing a force torque sensor with an attached square plate against multiple taxels. We then summed the total number of active taxels and divided the force by that number to get a force per taxel (or distributed force) value. We then took the calculated resistance of the active taxels, found the median, and correlated it with the force per taxel value. Figures 2.7-2.9 show the characteristic behavior of the sensor in four different scenarios: 1) the sensor is quickly loaded and unloaded multiple times to demonstrate the mechanical hysteresis of the fabric, 2) a load is applied at a slower rate of 1 $\frac{N}{s}$, 3) a load is applied at a faster rate of 5 $\frac{N}{s}$, and 4) the sensor is loaded to a specific value and unloaded to non-zero values multiple times (where the time dependence is represented by the colors transitioning from dark blue to yellow). Our main focus was to perform a simple calibration of the sensor and use

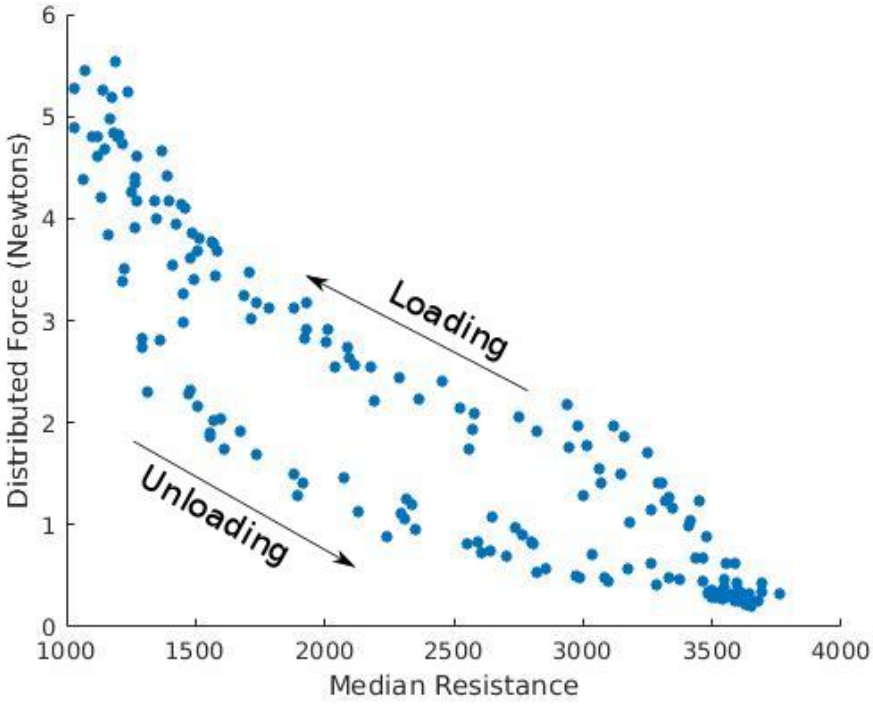


Figure 2.7: A plot showing the behavior of hysteresis in the material.

it on a single link robot arm for validation, so we did not attempt to compensate for hysteresis in this work. Methods can be employed such as those discussed in [30] to overcome the effects of hysteresis. Ignoring the effects of hysteresis, we used the linear calibration from Figure 2.8 for the loading curve and applied it to the sensor for testing, which was obtained by applying a force to a 4×4 square of taxels and taking the median estimated resistance value. We used the median of all the taxels instead of the average so that outliers would not dramatically affect the resulting calibration. The right figure in Figure 2.8 shows that calibration was fairly consistent for different rates of loading as well. Figure 2.9 shows the effects of hysteresis when the set of taxels undergo multiple loadings and unloadings. The taxels were first loaded to 4 N and unloaded to 1 N then loaded to 4 N again and unloaded to 2 N and finally loaded once again to 4 N, resulting in three loading curves.

Once the calibration was applied to the tactile sensor, we tested its accuracy and repeatability with the sensor array placed on a flat rigid surface. A 1.5 inch square of acrylic was pressed against the sensor while measuring the force with a force-torque sensor. The estimated force from each individual taxel was summed together to estimate a *total* applied force. Figure 2.10 shows

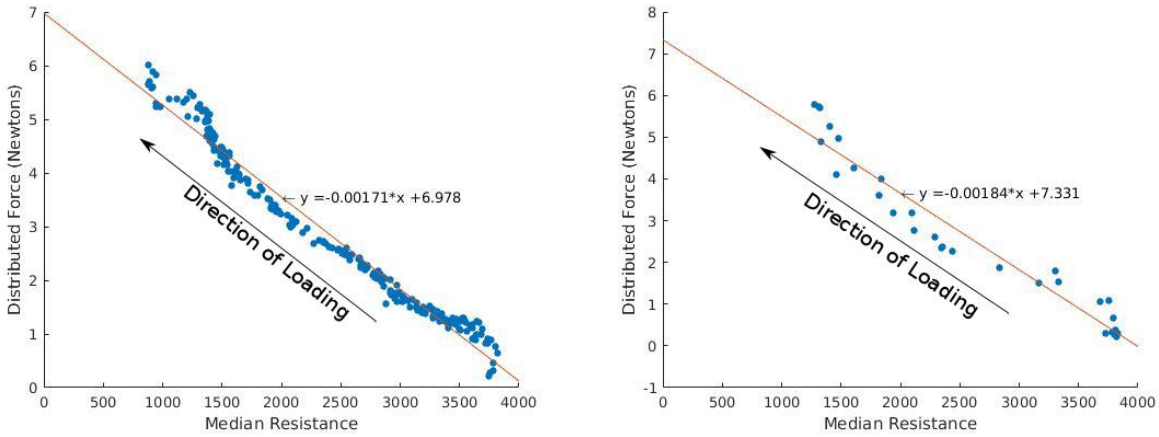


Figure 2.8: Left: A plot of the median of a group of taxel resistance values vs. force per taxel with a loading rate of $1\frac{N}{s}$. Right: A plot of the median of a group of taxel resistance values vs. force per taxel with a loading rate of $5\frac{N}{s}$.

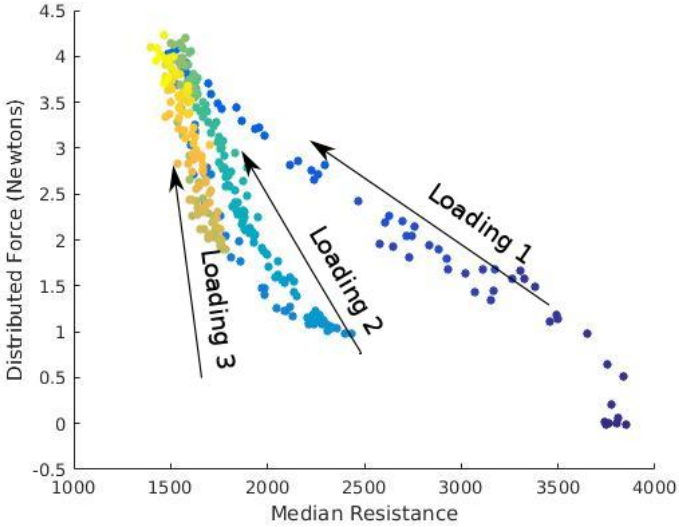


Figure 2.9: A plot showing repeated loading and unloading, where the sensor was unloaded to non-zero values before being loaded again.

the total estimated force compared to the actual applied force reported by the force-torque sensor. Although there is error and hysteresis, the accuracy for such large total forces is very encouraging.

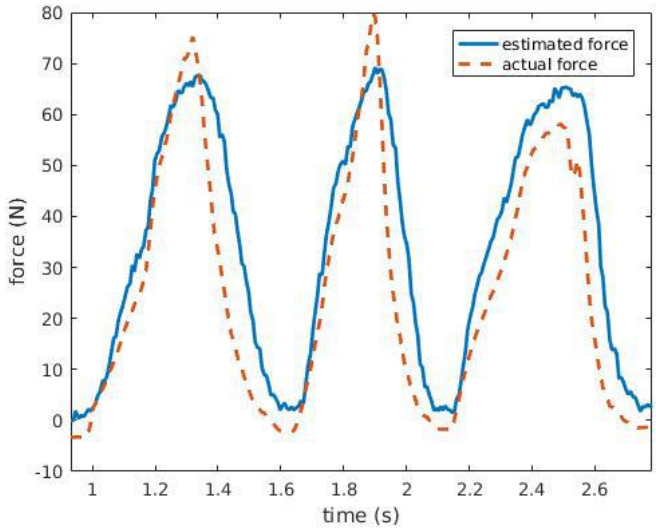


Figure 2.10: Sensor force estimate versus actual force when sensor is placed on a flat surface.

2.6 Tactile Sensor Applied to Soft Robot Link

Next, we wrapped the tactile sensor around the end of a single link inflatable robot. Figure 2.11 shows the sensor on the soft robot with a force torque sensor mounted on a stand. The robot was actuated so that it would press against the force-torque sensor. We pressed the robot link against the force-torque sensor multiple times while recording from both the tactile sensor array and the force-torque sensor. To compare the measurements from the tactile sensor array and the force-torque sensor, we summed the measured forces from each individual taxel to get a resultant force. Figure 2.12 shows how the tactile sensor array compared to the force-torque sensor. As was expected, the tactile sensor array measurements were noisier with the sensor wrapped around the soft robot limb than when it was tested on a flat, rigid surface. However, the overall accuracy and trends are quite good. Figure 2.12 shows the percent error of our estimated total force over time for the tactile sensor wrapped around the inflatable robot's limb. The large spikes represent the error due to the unmodeled effects of hysteresis occurring when the sensor is being unloaded. Any force estimates that were recorded as less than zero were assumed to be zero since the tactile sensor cannot be used to measure adhesive or negative forces. A video showing the collection of the data reported in Figure 2.11 is included as Supplemental Video 2 (https://youtu.be/qhI93MyNuFI).

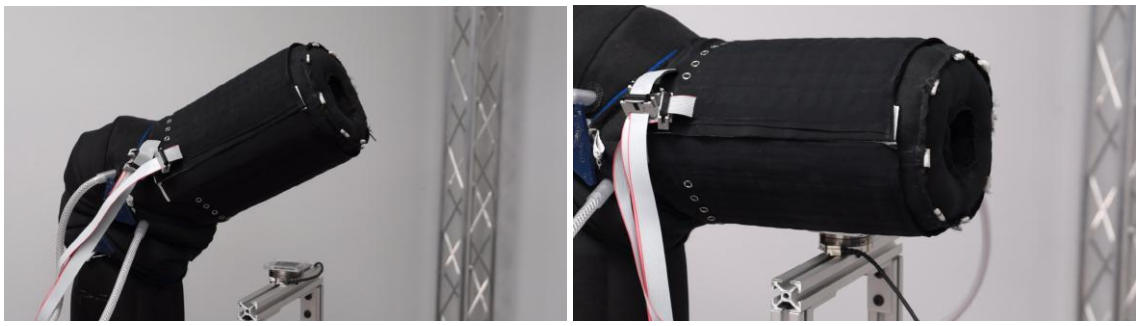


Figure 2.11: Images of tactile sleeve integrated onto a single link robot arm and the tactile sleeve being pressed against the force-torque sensor.

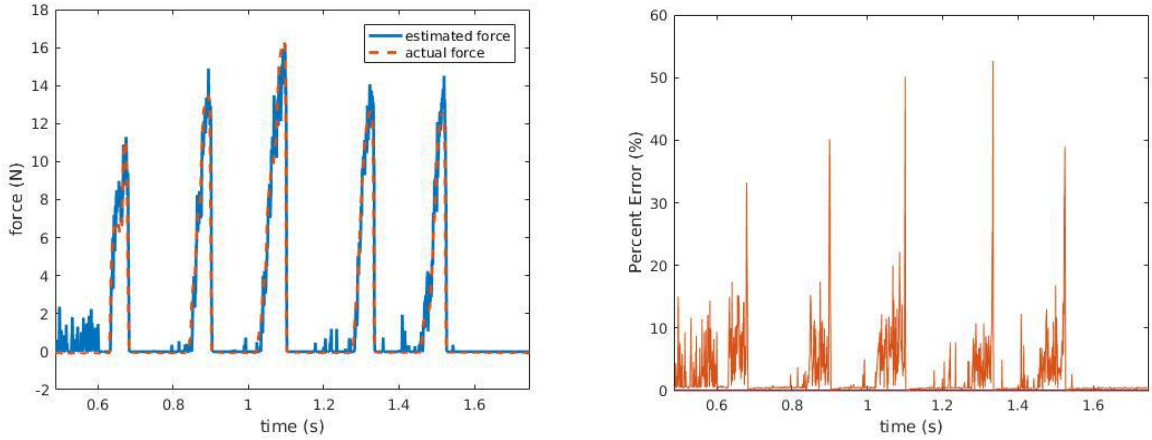


Figure 2.12: Left: The estimated force versus the actual force while the sensor is wrapped around an inflatable robot link. Right: This graph shows the percent error of the estimated force of the sensor on a single link robot.

Our sensor was developed as part of an SBIR Phase II program with NASA in collaboration with the startup company Pneubotics who has produced fabric-based, pneumatically-actuated soft robots, like the single joint shown in Fig. 2.11. In general, on our current soft robot hardware, we find that the main links of the robot do not bend or buckle and therefore tactile sensors on those sections would work well. However, sensors at the joints would obviously be deformed when the joint actuates and, similar to results in [13], would cause false positives if not compensated properly. However, because joint sensing for soft robots is still an open challenge as well, one possible application of our high resolution sensor is to integrate the sensors with the joints and use the data to learn models for joint deformation or configuration for soft robots.

2.7 Alternate Method of Cross-talk Compensation

2.7.1 Cross-talk Compensation with Additional Circuitry

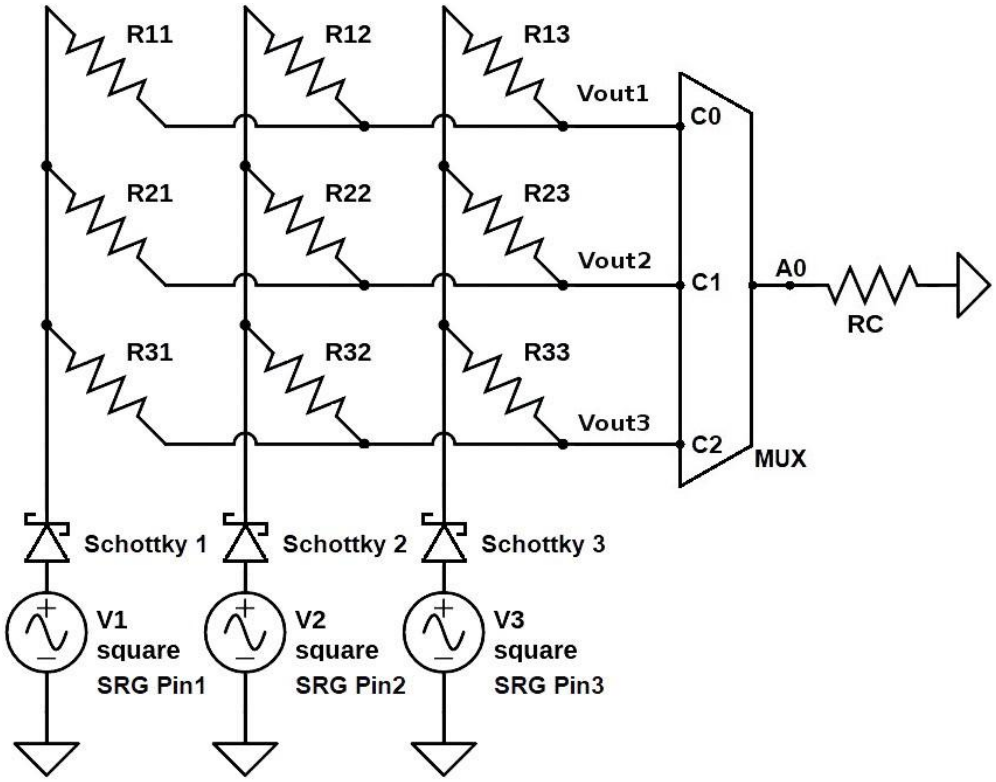


Figure 2.13: Alternative Circuit Diagram.

Electrical Closure of Parasitic Current Pathways

An alternative to the mathematical model compensation approach was developed in collaboration with Veronica Santos and Jimmy Pena of UCLA [29] to remove the source of crosstalk and is shown in Figure 2.13. The alternative approach prevents crosstalk by electrically closing parasitic current pathways. The components used were a multiplexer, a shift register, and one Schottky diode for each column in the sensor array. The multiplexer (Texas Instruments CD74HC4067) has a break-before-make switching mechanism that prevents crosstalk by breaking the connections to all readouts except one before reading [31]. All the sensor readout rows have an open circuit except

for the row being read, such that no parasitic current can flow in the open circuit directions. The constant value resistor R_C creates a voltage divider for the ADC to read from.

The shift register (Texas Instruments 74HC595) is used to simultaneously update the voltage values of the columns. Using a shift register means that the digital outputs to the columns of the sensor are synchronized. Updating the outputs simultaneously allows us to avoid reading from the sensor before the current is steady. As shown in Figure 2.13, the diodes are placed between the output of the shift register and the columns of the sensor. The diodes are used to prevent current back-flow into the shift register. As explained in the mathematical model, each sensor is modeled by a branching circuit or parasitic pathway; the crosstalk removal circuitry uses diodes to close off those branches and allows the output voltage of a sensor to be independent of the rest of the sensors in a given row. It should also be noted that the diodes used are Schottky diodes because they have a negligible voltage drop, thereby reducing loss in sensor resolution. The Schottky diodes used are 20 V, 1 A and have a voltage drop of approximately 0.01-0.02 V.

2.8 Validation of Cross-talk Compensation Methods

In this chapter, we presented two methods for crosstalk compensation. In this section, we present a final test to validate the effectiveness of the two compensation methods in terms of removing the effects of crosstalk. To show how crosstalk affects the circuit, we took a simple 3×3 tactile sensor array and applied a constant load to $\text{taxel}_{1,1}$ (which is the taxel in the first row and first column). An increasing force was then applied to $\text{taxel}_{1,3}$ (first row, third column). The goal of this test was to see how sensor output (either voltage or estimated resistance depending on the method used) of $\text{taxel}_{1,1}$ was affected by an applied load on $\text{taxel}_{1,3}$. In an ideal tactile sensor with no crosstalk, the reading for $\text{taxel}_{1,1}$ would be independent of the load on taxel $_{1,3}$ and would remain constant in this test. For a baseline comparison, Figure 2.14 shows the results when no crosstalk compensation is applied. The figure clearly shows that crosstalk occurs, even in this simple 3×3 sensor array.

The same test was then performed for the two crosstalk compensation methods in order to validate their effectiveness. Figure 2.15 shows the effect of applying a constant load to $\text{taxel}_{1,1}$ while increasing the force on $\text{taxel}_{1,3}$ for each of the two crosstalk compensation methods.

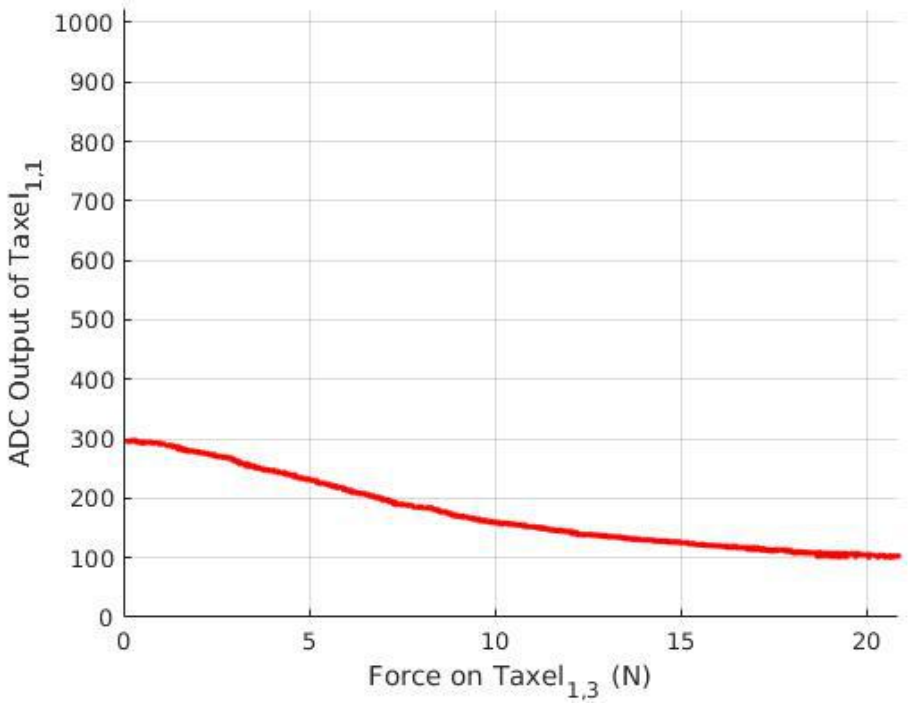


Figure 2.14: The effects of crosstalk without compensation.

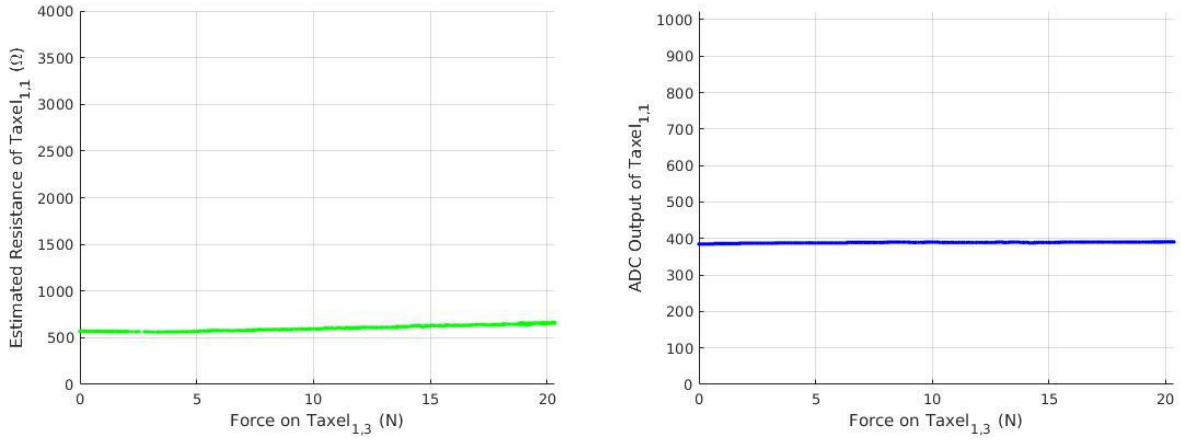


Figure 2.15: Left: Validation of crosstalk compensation for the method of modeling the circuit. Right: Validation of crosstalk compensation using the diode/multiplexer method.

As can be seen in Figure 2.15, the two methods of crosstalk compensation virtually eliminate the effects of parallel parasitic crosstalk that can be seen in Figure 2.14. Since the type of measurable unit differs for each compensation method, the Y-axis for each plot ranges from 0 to the full-scale range (FSR) of each method. The full-scale ranges for the resistance model and the

extra circuitry method are 0-4000 and 0-1024, respectively. These values would then need to be calibrated to force for actual applications. To compare the effectiveness of the two methods, we also calculate the slopes of the plotted lines in Figure 17. These slopes show the sensitivity of each compensation method to crosstalk during the loading. The slopes of the lines for both compensation methods stay relatively constant and near zero (especially compared to the slope of the uncompensated plot in Figure 16). Table 2.1 shows the percent change of each line, which was calculated by taking the slope of each line and dividing it by the FSR.

Table 2.1: Comparison of percent changes for the uncompensated graph and the two compensation methods.

Percent Change for Compensation Methods			
Force range	Uncompensated	Resistance Model	Extra Circuitry
0-20 N	1.0114	0.1182	0.0234
0-5 N	1.3726	0.0346	0.0745

Although both compensation methods were comparable for a 3×3 grid, we also performed the same test on our 11×27 taxel sensor that we attached to the soft robot in earlier tests. We only tested using the resistance model method (see 2.4), and the results are shown in Figure 2.16.

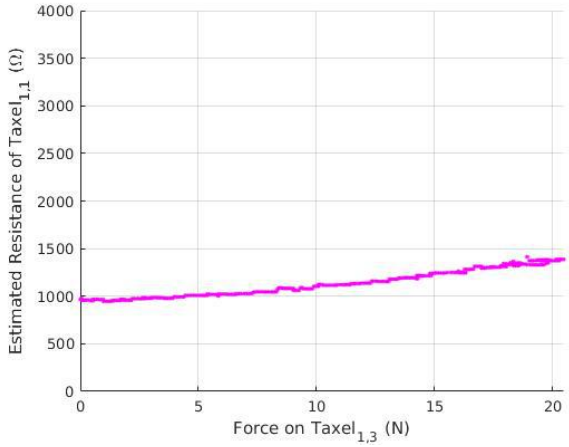


Figure 2.16: The effects of crosstalk on an 11×27 taxel sensor.

The crosstalk compensation is still most effective below 5 N when scaled, but as the force increases the crosstalk compensation begins to degrade. Interestingly, this only seemed to be true

for the resistance model version, and also occurred in the 3×3 version (Table 1). We expect that this is caused by the ADC resolution and the fact that the sensor is only powered by 5 volts. Since the voltage range for each taxel decreases as the number of taxels increases, an ADC with higher resolution is needed to capture smaller changes in voltage that are necessary for calculating the resistance accurately. Despite this limitation in our current hardware implementation, when the sensor is in use on soft robots, the total force will be distributed among multiple taxels where each taxel will experience a relatively small force as shown in Section 2.6. This means that the high resolution sensor in its current configuration can still estimate forces accurately for a certain range of forces per taxel.

From the results above we can see that the two methods are comparable methods for crosstalk compensation. These results also verify the effectiveness of the mathematical compensation method. This would be an appropriate compensation method if the user desires to reduce the amount of electronic components used in a circuit and thereby reduce price. If greater accuracy is required for higher forces, the diode method would be the method to implement.

CHAPTER 3. JOINT ANGLE ESTIMATION AND MACHINE LEARNING

3.1 Position Estimation methods

This chapter focuses on methods used to develop a soft-robot position estimation sensor and its characterization and calibration. Many sensing methods for soft robots have been explored in past work from various labs. [21] [25] [20] focused on similar physical phenomena that we also explore using based on light, magnetism, and resistance. Some of the approaches from work cited previously only involved bend angle estimation, but our overall goal was to estimate overall position and not just bend angle. Our sensor characterization efforts were focused on two main methods. The first method we explored attempted to use an analytical model developed from first principles to characterize the sensor's behavior. Because this method was not accurate enough for our purposes we turned to machine learning to develop a more accurate model that could account for noisy data and hysterseis. Three types of machine learning architectures were explored, including gated recurrent units (GRU's), long-short-term-memory cells (LSTM's), as well as convolutional neural networks (CNN's). We determine the best architecture to use for sensor characterization by comparing the estimation accuracy of all three architectures.

3.2 Sensor Design Ideas

To determine what type of sensor would best fit our needs, we experimented with a variety of prototypes. This prototyping process is not a significant contribution to soft-robot position estimation, but merely a way of determining what methods would be worth exploring further. We focused our designs on a single link pneumatically actuated soft-robot limb which will be referred to as 'the Grub' and is pictured in Figure 3.1.

The driving thought behind the various sensor designs and ideas was what physical changes in nature could we measure. Ideas included light, sound, magnetism, and change in resistance,

Figure 3.1: The Grub: a single link pneumatically actuated inflatable soft-robot.

among others. Magnetic fields can be measured using hall-effect sensors such as those found in magnetometers or magnetic rotary encoders. Light can be measured from LEDs emitting at various frequencies using photo-diodes. Changes in resistance occur in objects such as linear and rotary potentiometers and can be measured using voltage dividers and analog to digital converters (ADCs). These various measured phenomena can be directly correlated to changes in joint angles or link orientation. Figure 3.2 shows the various sensor concepts that we initially developed and explored.

The first sensor we experimented with was a photodiode and an emitter located at either end of a short section of PFA tubing. The idea behind this sensor was that the walls of the PFA tubing would reflect light towards the photodiode. As the bend radius of the tubing increased the amount of light that escaped through the walls would increase proportionally. Figure 3.3 shows a prototype of this type of sensor, and demonstrates the form of the sensor output as a function of

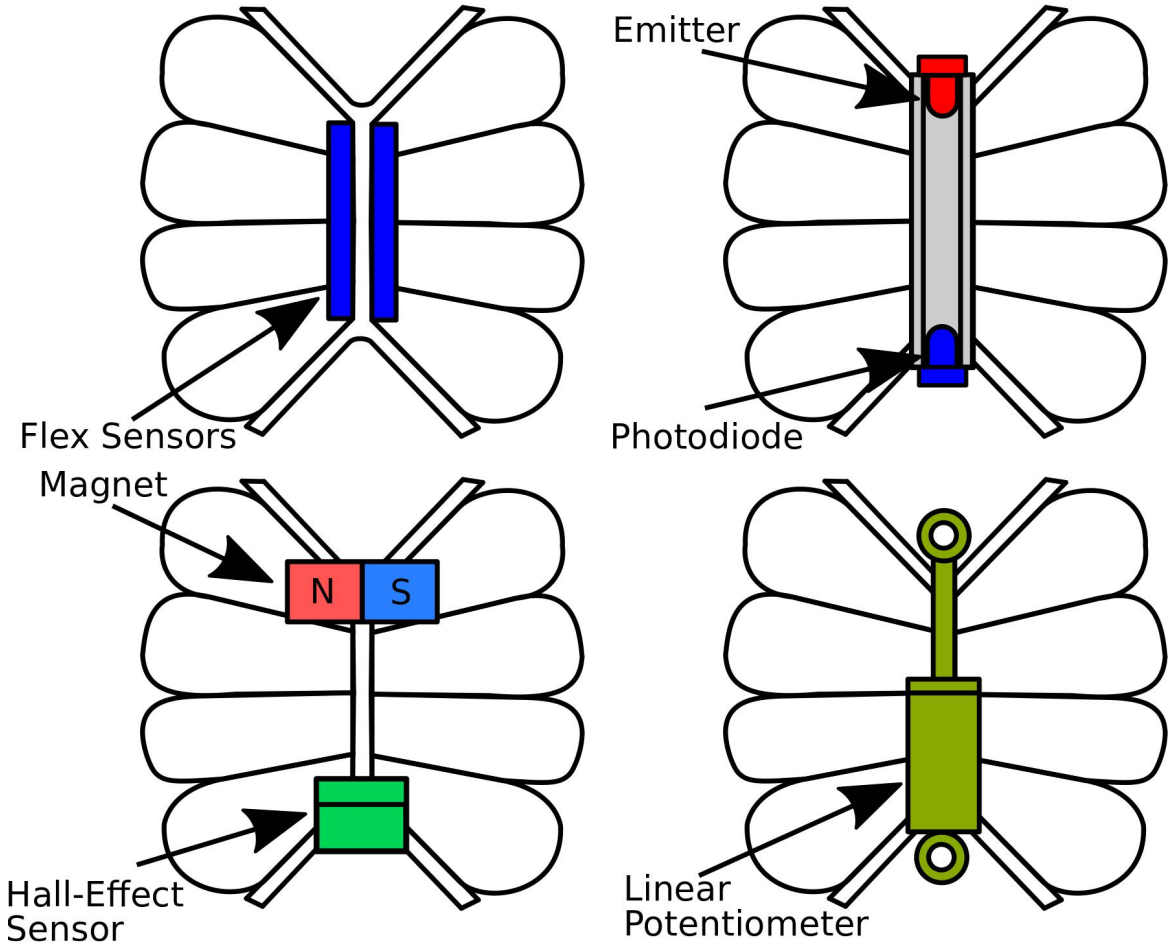


Figure 3.2: Top Left: Flex Sensor concept for joint angle estimation sensor. Top Right: Top Left: A photodiode and photo emitter connected using PFA tubing. Bottom Left: A magnet located near a Hall-effect sensor. Bottom Right: A linear potentiometer connected to top and bottom of grub joint.

bend angle. The main issue with this type of sensor was that it would be difficult to differentiate between positive and negative angles without redundant sensors. A second issue was that the output noise would increase substantially the closer the bend angle was to 0°.

The second sensing method we explored was a method based on magnetic fields (see Figure 3.2). A Hall-effect sensor that could detect a rotating field was be placed on one side of a soft robot's joint, and a magnet was then placed opposite the sensor. As the magnet rotated, the magnetic flux through the Hall-effect sensor would change. This resulted in a change in voltage and could be correlated to angle. This method seems like a viable solution and should be explored more in later work, but will not be further explored in this thesis. A potential downside to using this

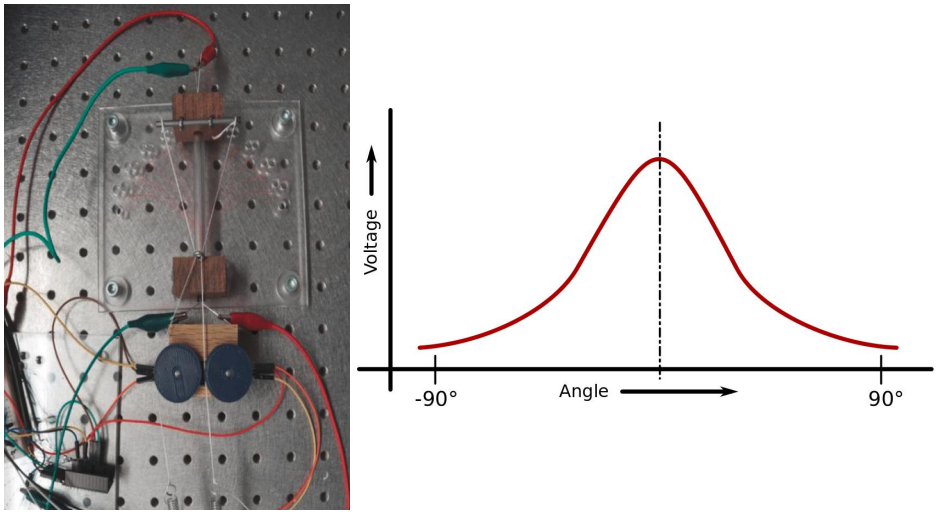


Figure 3.3: A rough prototype of a photodiode sensor as well as a curve representing the form of the photodiode and emitter sensor output.

type of sensors is that it requires that a very strong magnet be placed fairly close to the Hall-effect sensor. Such a large magnet could negatively affect nearby electronics, as well as have undesired interactions with external ferrous materials or current controlled pneumatic valves.

3.3 String Potentiometer Array

After exploring various types of sensors we concluded that we needed to develop a sensor that would give us more information than just rotation about a single axis. Because motion about the grub's joint was not constrained to rotation about a single axis, but could also rotate in off-axis directions and elongate. We needed to capture as much information in six degrees of freedom as possible: rotation about x, y, and z, as well as translation in x, y, and z. We also needed a sensor that was repeatable and would not be overly affected by mechanical wear. We settled on further experimenting with an array of string potentiometers attached to either side of a robot joint. If the string potentiometers were placed correctly, together they could all capture correlated, but complimentary information for the desired degrees of freedom.

The robot platform to which we attached the sensor array to is a single link, inflatable robot arm which we refer to as the Grub (Figure 3.4). The string potentiometer array for one side of a joint is constructed from five linear potentiometers placed side by side which are wired in parallel and mounted to the side of the robot's proximal link. String is wrapped around each of the

potentiometer slides. One end of the string is then tied to a spring for maintaining constant tension, and the other end is passed through an eye bolt mounted at the base of the joint. Each of the five strings are tied off to a row of pins that are set into a bracket and run orthogonal to the y-axis (see Figure 3.4).

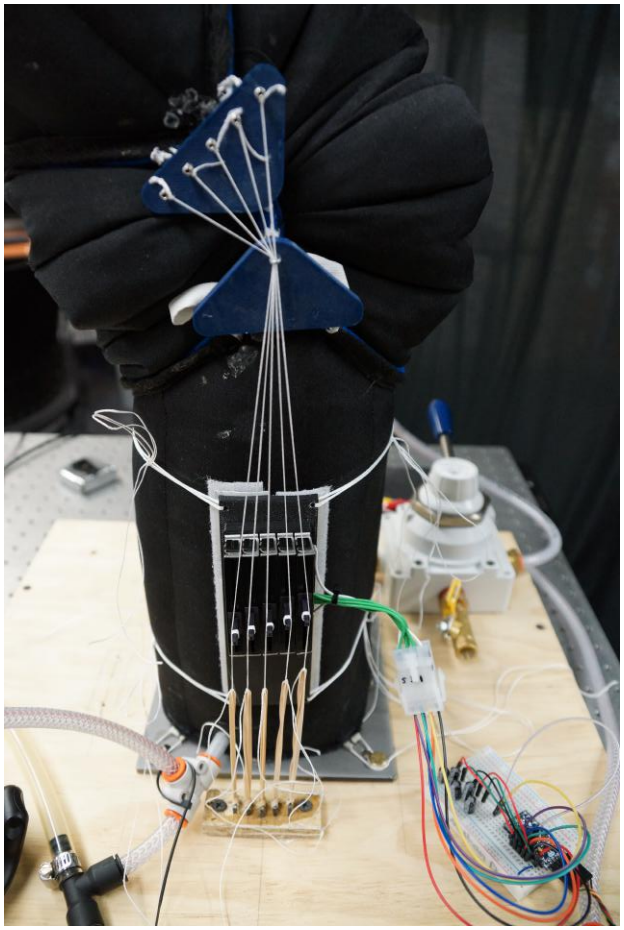


Figure 3.4: The string potentiometer array sensor.

A second sensor array is placed over the joint of the Grub opposite the side of the joint where the first sensor array was attached. As the Grub's distal link rotates about the main axis of rotation, the lengths of each of the strings will change. This change in length translates directly to a change in voltage in the linear potentiometers. The change in voltage is then read by an Analog to Digital Converter (ADC) and passed to a computer using an Arduino Pro Mini (Figure 3.5). The information is then extracted from the USB serial input and published to a node using the Robot

Operating System (ROS). The data is then available for any program to subscribe to it for use in collecting data for sensor calibration, or closed-loop feedback control.

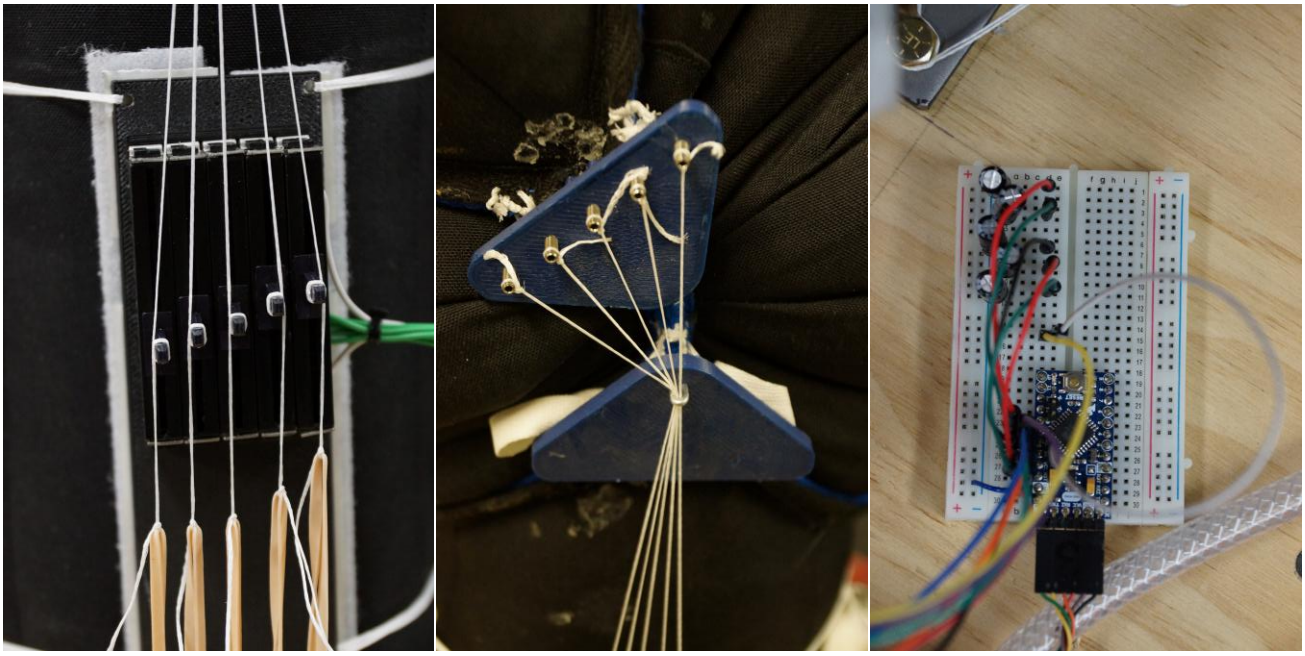


Figure 3.5: Left: Linear potentiometer array. Middle: Mounting bracket for strings. Right: Arduino Pro Mini used for Analog to Digital conversion.

The main reasoning behind the structure or geometry of the proposed sensor is that the redundant potentiometers can capture more of the motion characteristics of the Grub. As previously stated, the sensor will be able to capture motion information in the x, y, and z directions, and potentially the roll, pitch, and yaw of the Grub, and ideally even be able to estimate unmodeled disturbances along unactuated degrees of freedom. This will be especially useful, because unlike traditional rigid robots whose motion is constrained in all degrees of freedom except rotation about a pinned joint, the fabric-based, pneumatically actuated robot joints can have undesired movement in the x, y, z directions as well as undesired off-axis rotations. For all of our validation tests, we will be focused on the sensor's ability to estimate the x, y, and z positions of the end-effector

3.4 Analytical Models

Before we could correlate string lengths to motion for all degrees of freedom, we needed to develop a model that accurately demonstrates the relationship between changing lengths and changes in position. We focused on the parameterization of a continuum joint, which will be defined in section 3.4.1 for the development of an analytical model.

3.4.1 Continuum Joint Parameters

The parameterization of a three dimensional continuum joint can be found in [32]. The motion of the three dimensional continuum joint can be parameterized using three variables: u, v, and h. These form an axis-angle parameterization, where the axis of rotation lies solely in the xy plane and is defined by the vector formed by u and v, which does not pass through the origin, the variable h represents the arc-length of the continuum joint. The magnitude of $\overrightarrow{uv}$, represented by the variable ϕ, where $\phi = \sqrt{u^2+v^2}$, describes the magnitude of rotation that occurs about vector $\overrightarrow{uv}$ (see Figure 3.19). The h parameter is the arc-length of the center joint, which we will also be estimating for the analytical model. The homogeneous transformation matrix that describes the displacement and orientation of one end of a continuum joint relative to the base of the joint is shown in Equation 3.1,

$$g(u,v,h) = \begin{bmatrix} \sigma\tilde{v}^2+1 & -\sigma\tilde{u}\tilde{v} & \tilde{v}\sin\phi & -\sigma h\frac{\tilde{v}}{\phi} \\ \sigma\tilde{u}\tilde{v} & \sigma\tilde{u}^2+1 & -\tilde{u}\sin\phi & -\sigma h\frac{\tilde{u}}{\phi} \\ -\tilde{v}\sin\phi & \tilde{u}\sin\phi & \cos\phi & h\frac{\sin\phi}{\phi} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

$$g(u,v,h) = \begin{bmatrix} 1-\frac{1}{2}v^2 & -\frac{1}{2}uv & v & \frac{1}{2}vh \\ \frac{1}{2}uv & 1-\frac{1}{2}u^2 & -u & -\frac{1}{2}uh \\ -v & u & 1-\frac{1}{2}\phi^2 & h(1-\frac{1}{6}\phi^2) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

where $\tilde{u} = u/\phi$, $\tilde{v} = v/\phi$, and $\sigma = \cos\phi - 1$. This transformation matrix is singular when $\sigma = 0$. For the case when $u = v = \sigma = 0$, the transform can be approximated by Equation 3.2. For our tests

we used this approximation when ϕ was less than 1×10^{-8}. The continuum joint transformation matrix takes points expressed in frame f and expresses them in frame o as shown in Figure 3.6.

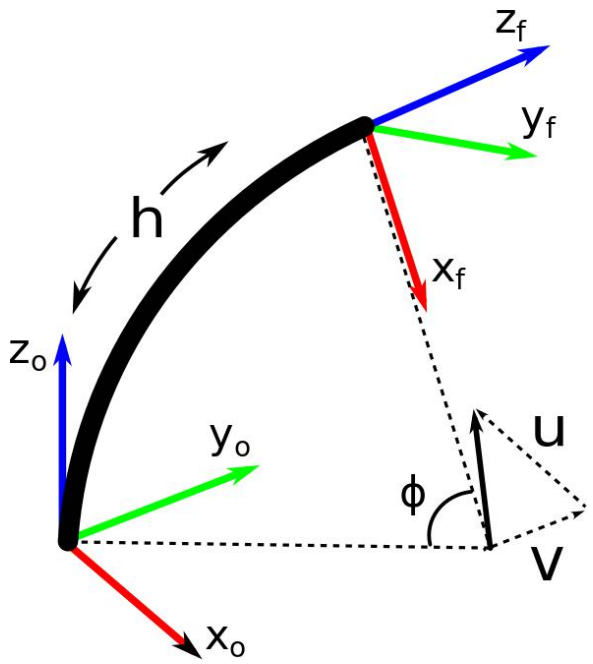


Figure 3.6: The continuum transformation is the transformation between frame f and o.

3.5 Simulation

Using this parameterization, our goal was to find Equations that represent the potentiometer string lengths from both string potentiometer array sensors as a function of u, v, and h, or in other words, find $L_n = f(u,v,h)$ where L_n represents the n^{th} string potentiometer length. We need this relationship eventually so that we can invert it and correlate string lengths to actual pose of the tip of the joint. Since we can only measure string lengths using the potentiometer array, we need to find some relationship between the string lengths and xyz position. In order to model the lengths of the string potentiometers, we used the continuum joint transformation representation to model the motion of the points where the potentiometer strings are tied off. The lengths of the strings are the distances from the tie off locations to some fixed point, Q. We will call the locations of the string tie-off locations p_n, where $p_n = \begin{bmatrix} x_n & y_n & z_n \end{bmatrix}^T$, as shown in Figure 3.7. These points are expressed relative to frame f. Using the continuum joint transformation from Equation 3.1, these

points can then be expressed relative to the o frame using the continuum transformation matrix, such that $p^o = g(u,v,h)p^f$.

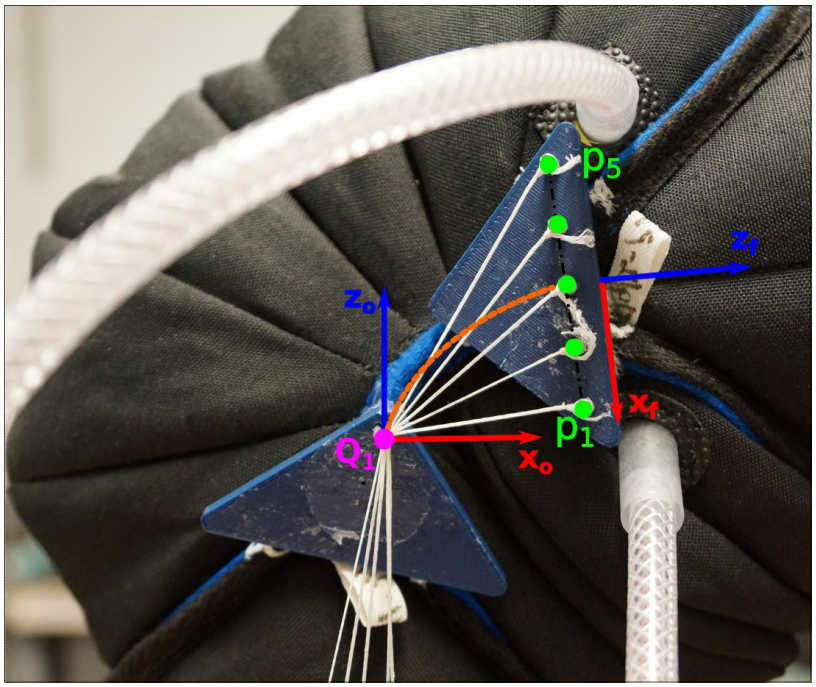


Figure 3.7: The locations of the string potentiometer locations and the intersection points of the two sensor arrays.

The length, L_n, is the euclidean distance between p_n and Q_1 for the sensor on the negative y_o side, and between p_n and Q_2 for the sensor on the positive y_o side, Thus $L_{1...5} = \|p_{1...5} - Q_1\|$ and $L_{6...10} = \|p_{6...10} - Q_2\|$. We assume we know the positions of p_n in the f frame. Since our goal is to find the relationship between L_n and u,v and h, we can use Equation 3.1 to develop a system of Equations that could be used to solve for u,v and h in terms of L_n.

We do not know p_n in frame o, but we directly measure the change in length of L_n. Substituting the vectors $\begin{bmatrix} x_n^o & y_n^o & z_n^o \end{bmatrix}^T$ and $\begin{bmatrix} x_n^f & y_n^f & z_n^f \end{bmatrix}^T$ in for p_n^o and p_n^f respectively, where p_n^o and p_n^f represent p_n in frame o and f, we get

$$\begin{bmatrix} x_n^o \\ y_n^o \\ z_n^o \\ 1 \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} & g_{13} & g_{14} \\ g_{21} & g_{22} & g_{23} & g_{24} \\ g_{31} & g_{32} & g_{33} & g_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_n^f \\ y_n^f \\ z_n^f \\ 1 \end{bmatrix} \tag{3.3}$$

We can show that $\hat{L}_n(u,v,h) = \sqrt{(x_n^o - Q_x)^2 + (y_n^o - Q_y)^2 + (z_n^o - Q_z)^2}$, where $\hat{L}_n$ represents the estimated length of the strings, therefore we can show that Equation 3.4 is true by substituting in the result from Equation 3.3.

$$\begin{aligned} \hat{L}_n^2(u,v,h) = \; & (x_n^f g_{11} + y_n^f g_{12} + z_n^f g_{13} + g_{14} - Q_x)^2 + ... \\ & (x_n^f g_{21} + y_n^f g_{22} + z_n^f g_{23} + g_{24} - Q_y)^2 + ... \\ & (x_n^f g_{31} + y_n^f g_{32} + z_n^f g_{33} + g_{34} - Q_z)^2 \end{aligned} \tag{3.4}$$

Solving for u, v, and h, analytically using these Equations is not possible since they cannot be isolated on one side of an Equation, because we have terms such as $\sin\phi/\phi$. Other ways to solve for u, v, and h, could involve finding the roots of the Equation, or optimization methods. The draw back of these methods are that they yield an approximate result. Since the Equation does not have a tractable solution we decided to employ optimization methods.

Our first attempt was to use MATLAB's *fmincon* function. This allowed us to apply constraints to the optimization since we knew that u and v would not exceed $\pi/2$ rad or 0.3 rad respectively, and that h would not exceed 8 cm. A residual was formed by using the difference between L_n^2 and $\hat{L_n}^2$. The cost function was defined as

$$\min_{u,v,h} \sum_{n=1}^{10} (L_n - \hat{L_n})^2 \tag{3.5}$$

Using *fmincon*, we were not able to get an accurate result. At each iteration the optimization would get trapped in a local minimum and quit. To remedy this we switched over to a global optimization method that allowed for a global search within the lower and upper bounds known as particle swarm optimization. To test the particle swarm optimizer, we generated a trajectory for vector $\overrightarrow{uv}$, and used the resulting trajectory to find the end-effector position at any time t. We then used the particle swarm optimizer to minimize the cost function by changing u, v, and h.

We first tested using simulated sensor lengths without any noise. Figure 3.8 shows the estimated end-effector position compared to the end-effector ground truth position. The next step was to test how well the optimization could solve for u, v, and h when noise was introduced. Using noise with a standard deviation of $\sigma = 0.1$ cm on the simulated lengths, we were able to estimate the error on u, v, and h to within 0.03 rad, 0.003 rad, and 0.03 cm, respectively. Figure 3.8 shows the estmiated end-effector position compared to the end-effector's ground truth position. Since these results were promising we decided to test the analytical model using the particel swarm optimizer on the physical robot for live testing.

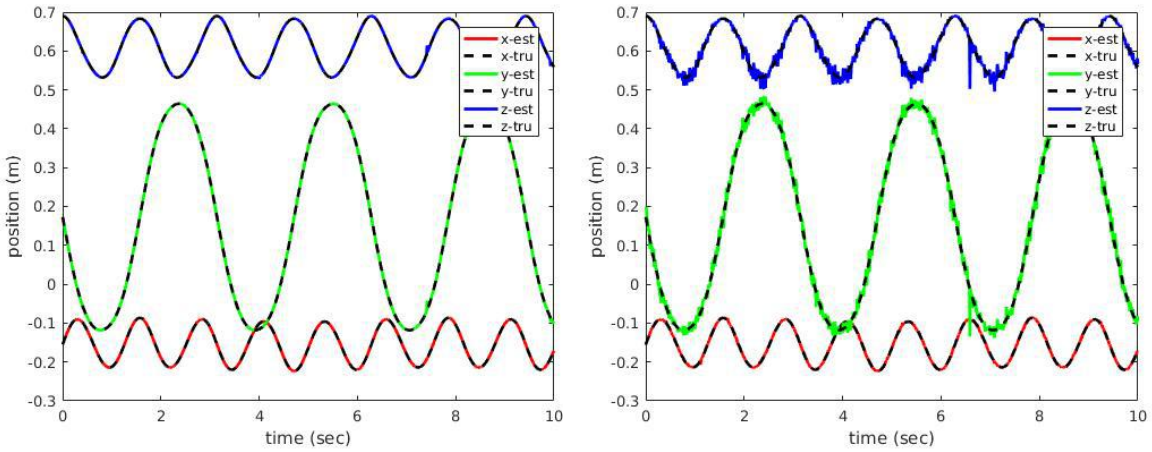


Figure 3.8: Left: Simulated analytical model *xyz* estimates compared to *xyz* truth without noise added, where *x*, *y*, and *z* are represented by red, green, and blue lines respectively. Right: Simulated analytical model with noise added *xyz* estimates compared to *xyz* truth.

3.5.1 Testing Analytical Model

In order to test the analytical model on real hardware, we required a method of reporting the ground truth position of the grub's end-effector. We used the HTC Vive virtual reality system to report the true *xyz* position and frame orientation of a Vive tracker fastened to the Grub's end-effector relative to a base frame. A second Vive tracker marks the position of the base frame and is located adjacent to the Grub's base, offset in x and y. Figure 3.9 shows the locations of the Vive trackers and Figure 3.10 shows the positions of the coordinate frame used for the forward kinematics of the Grub.

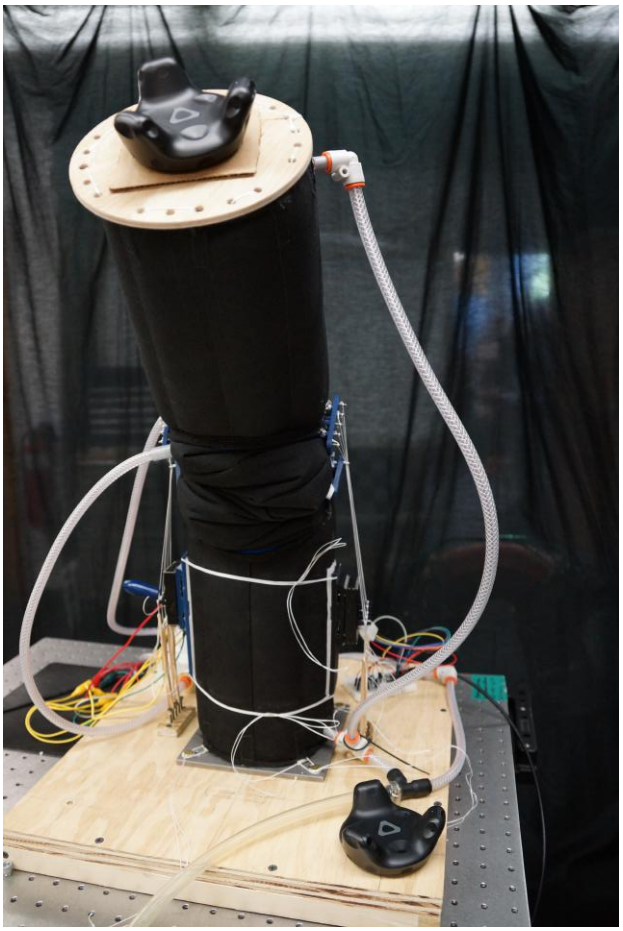


Figure 3.9: The locations of the Vive trackers on the end-effector and the grub base.

Four coordinate frames are attached to the grub at key locations as shown in Figure 3.10 for modeling the forward kinematics of the grub. Frame O_1 is attached to the Vive tracker on the grub base, frame O_2 is attached to the center of the proximal link at the base of the continuum joint, frame O_3 is located at the top of the continuum joint just below the distal link, and finally frame O_4 is located at the end-effector position where the second Vive tracker is attached.

The transform between frame O_2 and O_3 is the transform defined previously in Equation 3.1. The transforms from O_1 and O_2 and between O_3 and O_4 are shown in Equations 3.6 and 3.7

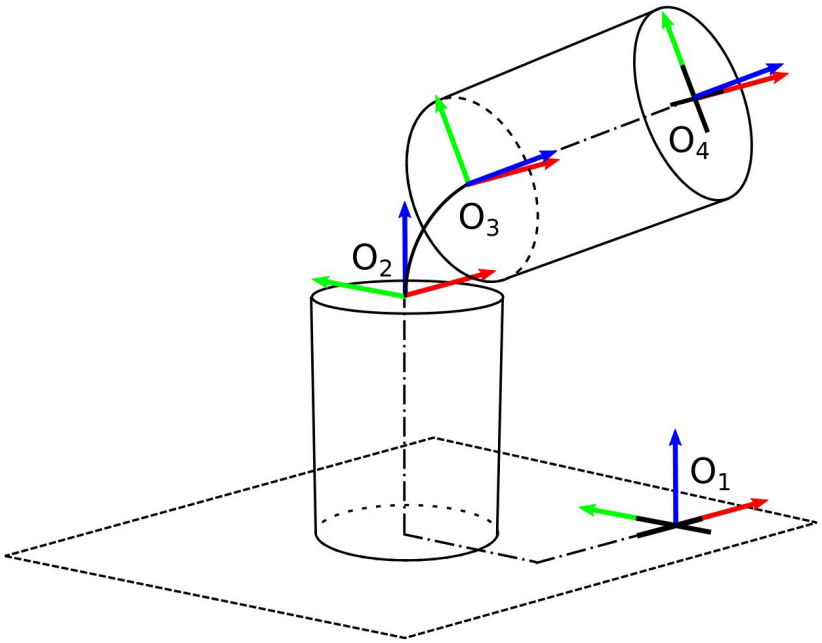


Figure 3.10: The frame locations used for the forward kinematics of the Grub.

respectively, where the values were obtained from manual measurements.

$$T_{O_2}^{O_1} = \begin{bmatrix} 1 & 0 & 0 & -0.155 \\ 0 & 1 & 0 & 0.173 \\ 0 & 0 & 1 & 0.320 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

$$T_{O_4}^{O_3} = \begin{bmatrix} 1 & 0 & 0 & 0.00 \\ 0 & 1 & 0 & 0.00 \\ 0 & 0 & 1 & 0.320 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

Using these transforms we can show that the transform between O_4 and O_1 is Equation 3.8.

$$T_{O_4}^{O_1} = T_{O_2}^{O_1} T_{O_3}^{O_2} T_{O_4}^{O_3} \tag{3.8}$$

Using Equation 3.8 a point relative to frame O_4 can be expressed in frame O_1 by the transformation $p^{O_1} = T_{O_4}^{O_1} p^{O_4}$.

The next step was to convert the ADC values reported from the string potentiometer sensor to lengths in meters. We found that the length L_n was a function of the total string length from tie off location to linear potentiometer slide and simple geometry of the linear potentiometer array as shown in Figure 3.11.

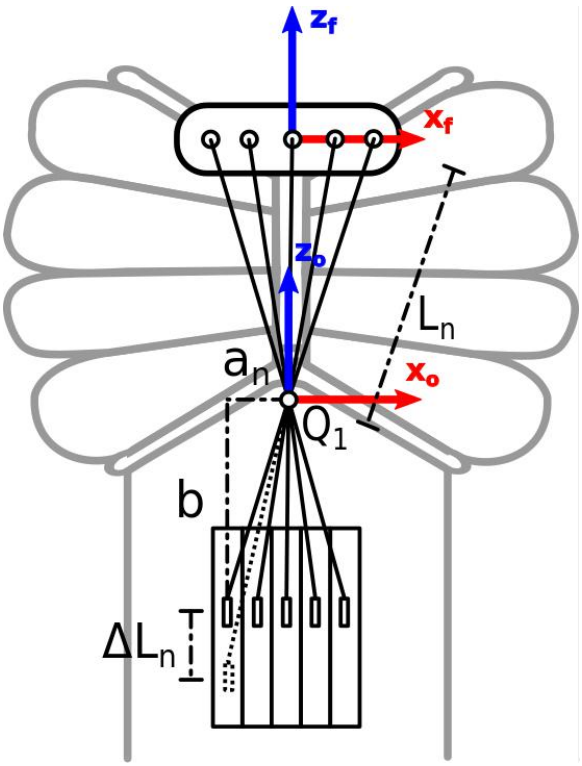


Figure 3.11: The sensor geometry used for converting ADC values to lengths.

A geometric relationship for the lengths is shown in Equation 3.9. The variable a_n represents the horizontal from point Q_m for each potentiometer slide, and variable b represents the y-distance from point Q_m for the initial starting positions of each slide. K_{ADC} is the unit conversion from ADC values to meters, and ADC_n represents the ADC value of the n^{th} potentiometer voltage output. $L_{n,total}$ is the total length of the string measured from tie-off location to potentiometer slide, and ΔL_n represents the change in distance of the potentiometer slide from its original starting position.

$$L_n = L_{n,total} - \sqrt{a_n^2 + (b - \Delta L_n)^2}, \quad \text{where} \quad \Delta L_n = K_{ADC} ADC_n \tag{3.9}$$

After calculating the lengths given ADC values, we ran a test with the analytical model in simulation. The algorithm used to estimate x, y, and z using the analytical model is shown in algorithm 1.

Algorithm 1 Analytical Model Procedure

1: Get initial ADC values
2: **while** True **do**
3: Subtract initial ADC values from each ADC reading;
4: Convert ADC to length in Meters;
5: Pass lengths into Particle Swarm Optimizer;
6: Minimize residual of Equation 3.4;
7: Use estimated u, v, and h in homogeneous transform between frames O_3 and O_2;
8: Calculate $T_{O_4}^{O_1}$;
9: Pull x, y, and z from fourth column of $T_{O_4}^{O_1}$ to obtain end-effector position;

Once the estimated end-effector position was found, we compared it to the real position to determine the accuracy of the analytical model. Figure 3.12 shows a plot of ground truth compared to the estimate. The red line represents the truth data for x, y, and z, and the blue line represents the estimate. The left figure shows how noisy the estimation using the particle swarm optimization on the physical robot actually is. Instead of an error of 5×10^{-5} m we obtained in simulation, we were getting errors of 0.4 m. We low-pass filtered the data to see how close of an estimate we had if we eliminated as much noise as possible. The figure on the right shows the data low-pass filtered to show that the estimate was not just random noise and that the majority of the position estimates were relatively close to the actual position.

We could have focused more on decreasing the noise of the particle swarm estimate, but felt that our effort would be better spent elsewhere because the analytical model would not be accurate enough even if the noise was reduced. There are a few reasons why the analytical model did not perform well. We observed that as the bladders on the grub inflated and deflated, they would do so at different rates. This caused the arc length of the joint center to increase and decrease depending on the direction of rotation. This introduced path dependent behavior known as hysteresis, which is difficult to model geometrically in an accurate way. Other reasons why the analytical model breaks down could include manufacturing defects in the fabric structure, and geometric measurement

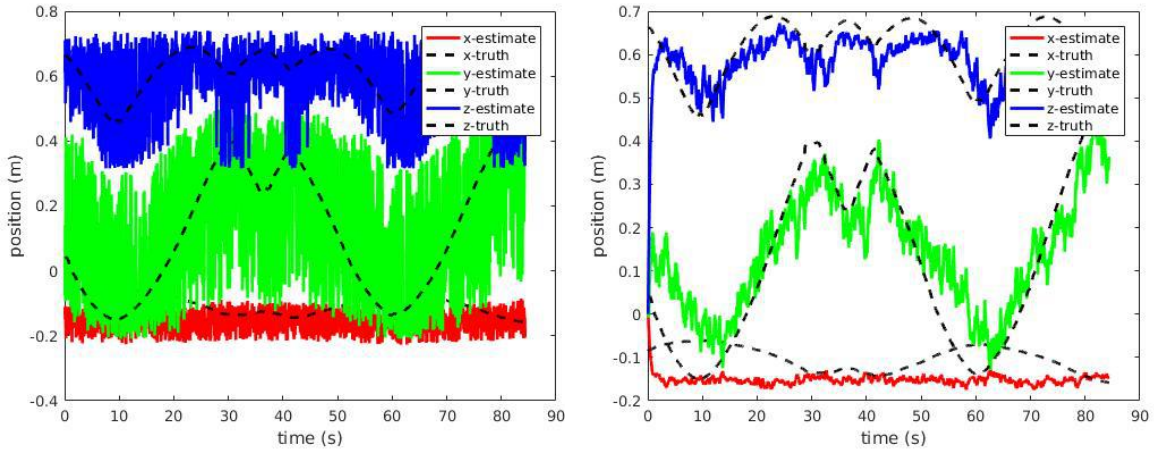


Figure 3.12: Left: Analytical model *xyz* estimates compared to *xyz* truth, where *x*, *y*, and *z* are represented by red, green, and blue lines respectively. Right: Analytical model with low-pass filtered *xyz* estimates compared to *xyz* truth.

error inherent to manual measurements used for estimating model parameters. It is difficult, if not impossible, to manufacture fabric based structures to a high degree of accuracy. This means that there are unpredictable effects the structure will have on motion that cannot easily be modeled. Another method is required that can more accurately account for the uncertainties in measurements and construction of the robot.

3.6 Machine Learning

Because the interactions between the soft robot and the sensor geometry are complicated to model, and certain characteristic responses to motion cannot be anticipated, an alternative to an analytical model based on first principles was machine learning. Machine learning has been shown to be very effective in modeling complex behaviors of objects, as well as developing algorithms that capture complexity that human-built algorithms are not able to. We decided to use machine learning to develop an approximate model of the sensor. Machine learning in its basic form is a method of optimization that uses a network of weights to approximate functions or models. We used an open-source machine learning framework developed by Google called *TensorFlow* [33] to program our various neural network architectures. We first look at the simplest type of neural network, where we simply stack fully-connected layers together (section 3.6.1).

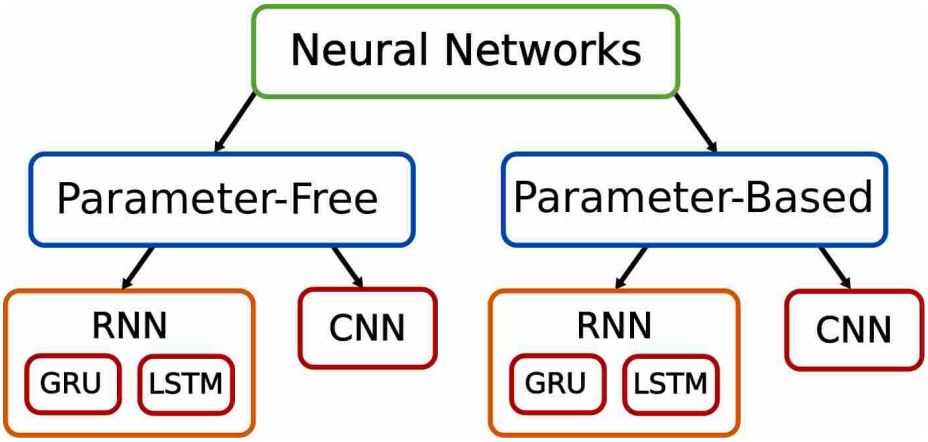


Figure 3.13: A web diagram showing the breakdown of machine learning where we train three neural network architectures using parameter-based methods and parameter-free methods.

Because fully-connected layers were not able to estimate *xyz* position accurately with raw ADC data, we transitioned to experimenting with other types of neural network architectures such as recursive neural networks (RNN's), including GRU cells and LSTM cells (sections 3.8.5 and 3.8.5), convolutional neural networks referred to as CNN's (section 3.8.5). To verify that these architectures were even feasible alternatives to fully-connected layers, we first tested them on simulated data. Once we verified their trainability through simulation we trained them on raw ADC values that resulted from moving the Grub platform around. We also explored whether parameter-based learning, learning that forces a learned model to conform to a specific parameterization, would result in better position estimation than parameter-free learning, and which architecture results in a more accurate estimation when disturbances are added to the training data. Figure 3.13 shows the break-down of machine learning methods that we explored into smaller components for testing and validation.

Parameter-Free Learning

One hypothesis we wanted to test with machine learning was whether or not parameter-based learning for soft robots is more accurate than parameter-free learning. Parameter-free training is when a neural network learns the end-effector position without any kind of robot parameters, such as link lengths, offsets, or forward kinematic models. Parameter-free learning will bypass the robot parameters and simply try to estimate end-effector position directly from the given inputs

from the joint estimation sensor. In our specific case, a parameter-free network will take in the ten inputs from the two sensors and learn how to estimate the end-effector position in Cartesian space.

Parameter-Based Learning

Parameter-based learning is an attempt to learn a robot's kinematic parameterization which is then used to calculate the robot's end-effector position. The outputs from the neural network are passed into a forward kinematic model that will be used to estimate the *xyz* position of the end-effector expressed in the base frame, $T_{O_2}^{O_1}$. Three models were used for parameter-based learning, starting with the simplest parameterization. Model complexity was then increased to determine at what point increased model complexity would not increase estimation accuracy. The subsections below contain the descriptions of the neural net architectures that we used.

3.6.1 Fully-connected Layers

Our initial approach for developing a model of the sensor for joint estimation was based on a simple neural net using only fully-connected layers. The ADC values from the sensor were passed into the network as well as ground truth values representing translation in the x, y, and z directions making this a supervised learning approach. The diagram in Figure 3.14 shows a simple representation of this network. The ADC values from one or both of the joint sensors are fed into the network. The form of this network is $Ax = b$, where A is the fully-connected layer, x is the array of inputs, and b is the array of outputs. If more layers are added, such as is shown in Figure 3.14, then the output from the previous layer, b, becomes the input, x, for the following layer, and so on.

We first ran tests using three fully-connected layers. Our inputs for the network were the ADC values collected from the string potentiometers. The initial xyz estimation error was an average of 0.09 m for the total xyz distance. Constructing a deeper network by increasing the total number of fully-connected layers seemed to have little to no effect on decreasing the end-effector estimation error. When we increased the number of fully-connected layers to 11, the estimate was even worse with an average error of 0.148 m. It appears that the fully-connected layers cannot

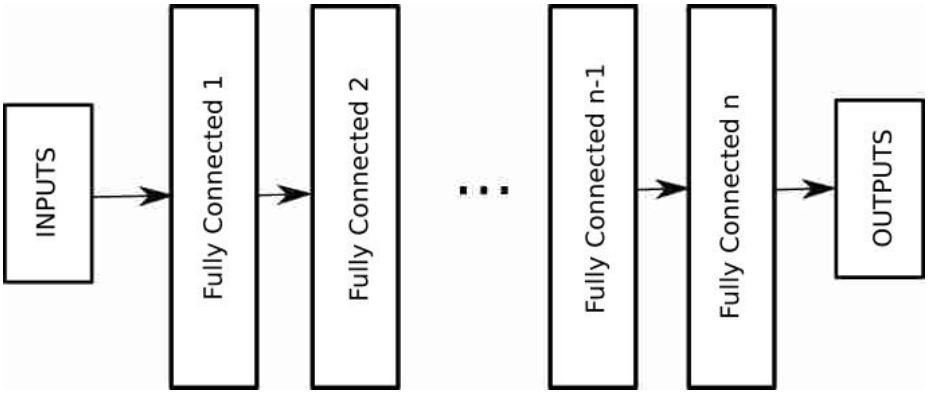


Figure 3.14: A simple multi-layer network.

represent the variability in the structure of the soft robot limb effectively. This could be due to the presence of hysteresis in the angular rotation dependent on the direction of rotation.

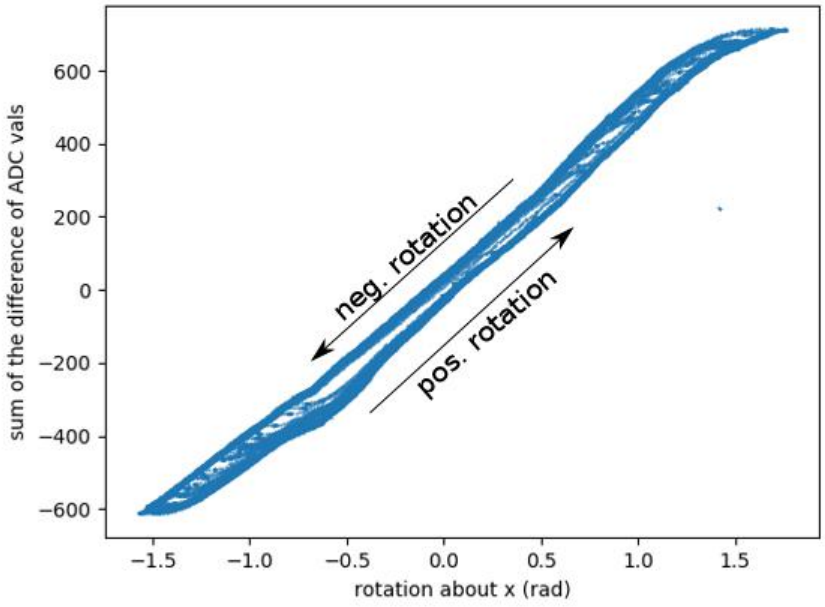


Figure 3.15: Hysteresis due to positive and negative rotation.

Fully-connected layered networks are basically function approximators. Since hysteresis is present in the system, the fully-connected layer network does not seem to be able to estimate the desired end-effector position accurately enough for our needs. To show the effects of hysteresis, we took the difference between the five ADC outputs from one of the sensors and summed the

resulting four numbers together in order to visualize a single curve as a function of rotation about the x-axis. Figure 3.15 shows how the output of the joint sensor is dependent on the direction of rotation. When rotating in the negative direction the path lies along the upper curve. When rotating in a positive direction the path lies along the bottom curve. This is caused by inconsistencies in the construction of the soft robot arm due to its hand made construction and the use of non-rigid materials.

3.6.2 Recursive Neural Networks

Because hysteresis is present in the robot movement, another method for modeling the sensor needed to be found. Since hysteresis is time dependent, or path dependent, we needed to use an architecture that has a way of keeping track of information from past inputs. Recursive Neural Networks, or RNN's, are neural networks that depend on previous outputs. An input vector, x_t, is passed into the RNN, and a state vector, h_{t-1}, is also passed in. A state vector, h_t is then passed out of what is called an RNN cell. These cells contain gates that determine how much of the previous input the RNN cares about and how much of the output vector is relevant to estimation accuracy.

RNN cells can be cascaded such that the output state vector, h_t, can be passed in to a consecutive cell, and so on. Figure 3.16 shows the architecture of cascaded RNN cells. Initial inputs are rearranged into a one-dimensional array and passed through a fully-connected layer to form the input vector x_t. The state vector and the gate vectors of the RNN are automatically formed using functions from *TensorFlow*.

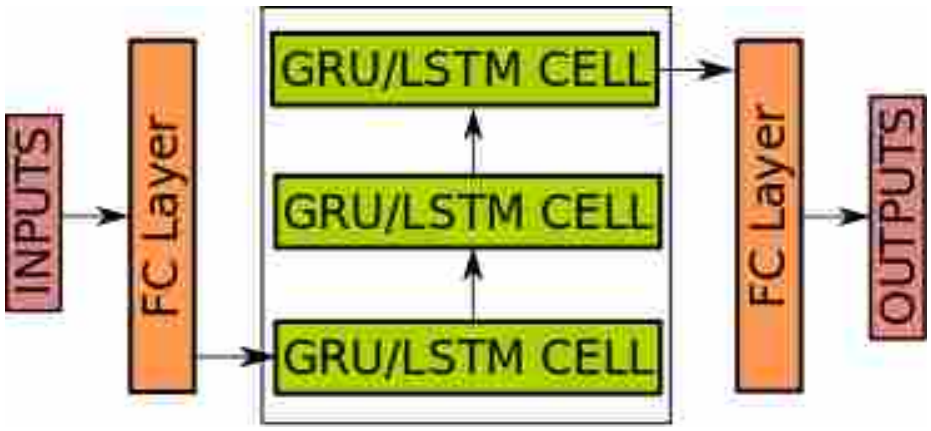


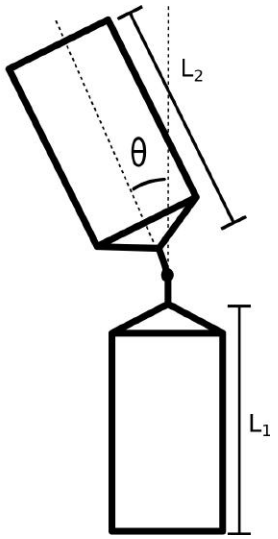
Figure 3.16: A diagram showing he the architecture of recursive neural network is set up.

3.6.3 Parameterized Models

The following sections describe the parameter-based models that we used for testing our parameter-based machine learning. We first discuss a simple pin joint model, followed by a two dimensional continuum joint model. The last parameter-based models we explore are based on a three dimensional continuum joint. We start by using a simple three-parameter 3-D continuum joint model, followed by two more variations of the 3-D continuum joint each with an increase in number of parameters used.

Pin Joint

The simplest model for representing a single link robot is a pin joint model. This results in pure rotation of the end-effector about a central position. The *xyz* location of the end-effector, if positive rotation is defined about the positive x-axis, can be expressed as shown in Figure 3.17.



$$x = 0$$
$$y = -L_2 \sin\theta$$
$$z = L_1 + L_2 \cos\theta$$

Figure 3.17: The parameters of a pin joint model.

This model was explored because pin joints have been used in past work on soft-robots as an approximate model for constant curvature joints, and we wished to explore how end-effector position was affected by this assumption.

2-D Constant Curvature

A slightly more complicated model uses constant curvature to model the rotational behavior of the black grub. As the grub's distal link rotates about the x-axis, the central span of the joint forms an arc. As the magnitude of θ increases the radius of curvature decreases. This is demonstrated on the physical Grub in Figure 3.18. Figure 3.18 shows the geometry used for this model. One can also observe that there is no static joint center. The center of rotation can translate in y

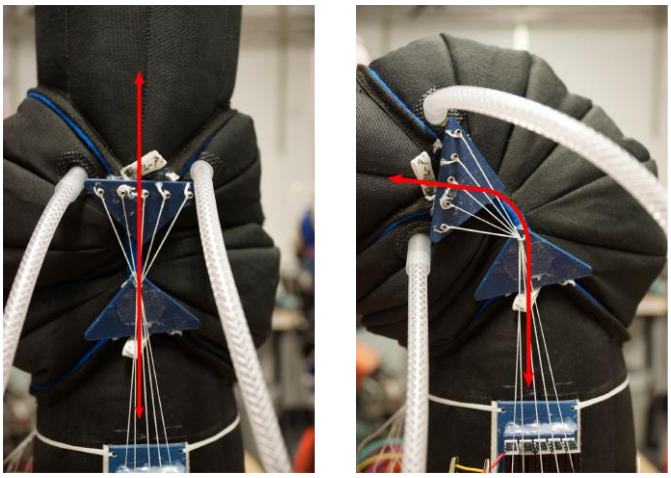
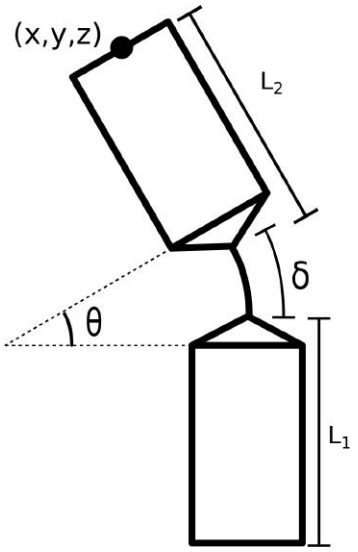


Figure 3.18: Left: Red line shows curvature of arm when vertical. Right: Red line shows curvature when arm is horizontal.

and z. The xyz position of the end effector can be expressed as shown in Figure 3.19.

3-D Continuum Joint Model

Three variants of this model were be explored. These models are based on the transformations from Equation 3.8 and Figure 3.10. For the first variant of this model, the inputs from the neural network were the transformation's inputs u, v, and h. $T_{O_2}^{O_1}$ and $T_{O_4}^{O_3}$ remain constant with the xyz offsets and link lengths as hand-measured estimates.



$$x = 0$$
$$y = \delta - \sin\frac{\theta}{2} - L_2 \sin\theta$$
$$z = L_2 + \delta\cos\frac{\theta}{2} + L_1\cos\theta$$

Figure 3.19: Parameters of a 2D continuum joint with constant arc length δ.

The second variant attempted to estimate the link lengths of the robot instead of assuming that the measured link lengths are accurate. Instead of depending on hand measurements, the neural network will determine what link lengths should be used. This does not mean that the learned link lengths will be constant values, but this is not necessary for accuracy.

Aside from learning parameters u, v, and h, the third and final variant will attempt to learn the link lengths as well as all the translational xyz offsets of $T_{O_2}^{O_1}$ and $T_{O_4}^{O_3}$. This should be able to get rid of error due to approximate measurements, as well as offsets that were not included in the homogeneous transformations, such as possible offsets where the Vive tracker is located at the end-effector. For example, the Vive tracker may have offsets in the x and y directions of the O_4 frame that were not accounted for. The purpose of these three continuum joint model variants was to see at what point, if any, does an increase of parameters stop increasing the accuracy of the estimated end-effector position to determine if learning more parameters is better than learning fewer parameters.

3.7 2-D Simulation and Validation Results

Before training the neural networks on raw data, we wanted to verify the trainability of the neural network using simulated data, similar to how we verified the analytical model through simulation before testing on the Grub. We also wanted to see how much error we could expect when the motion of a continuum joint was approximated using a pin joint model. We generated 2D simulation data using the 2D continuum joint model from Figure 3.19 using parameters θ and ϕ from Figure 3.19 for rotation and translation to see how accurate our estimate could be in simulation. We trained an RNN network with GRU cells on the generated 2-D simulated data to make sure that our neural network architectures could be trained on data from this type of sensor, before training on data that was noisy and suffered from hysteresis. Figure 3.20 shows the form of the generated input data consisting of simulated ADC values.

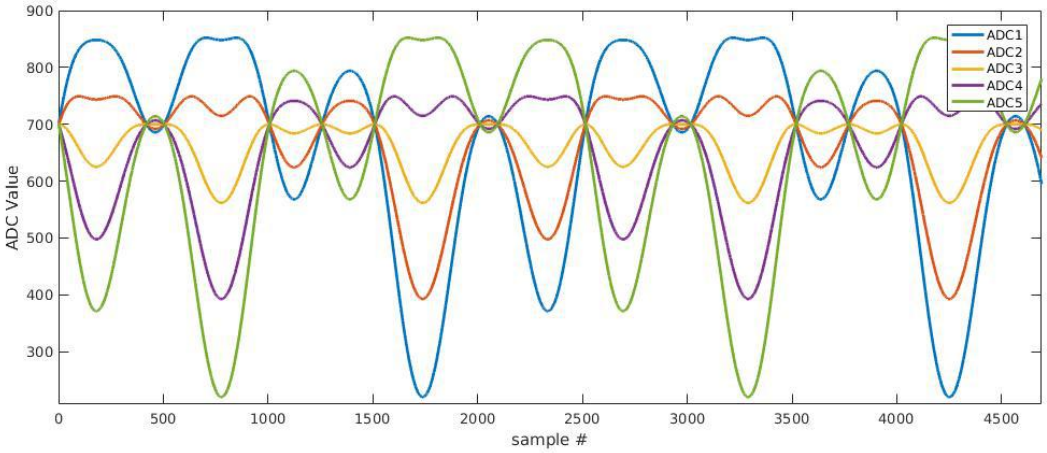


Figure 3.20: Simulated data used to verify neural network training.

To verify the accuracy of the machine learning estimate, we took a snippet of the testing data and ran it through the neural network every few training iterations to see how the estimate lined up with the ground truth values. This allowed us to see how accurate the neural net was at estimating *xyz* position using simulated data as the network was being trained. This also helped us verify that the neural network was not overfitting the training data. Figure 3.21 shows how well the estimated *y*-position and the actual simulated *y*-position lined up using the simulation data shown in Figure 3.22.

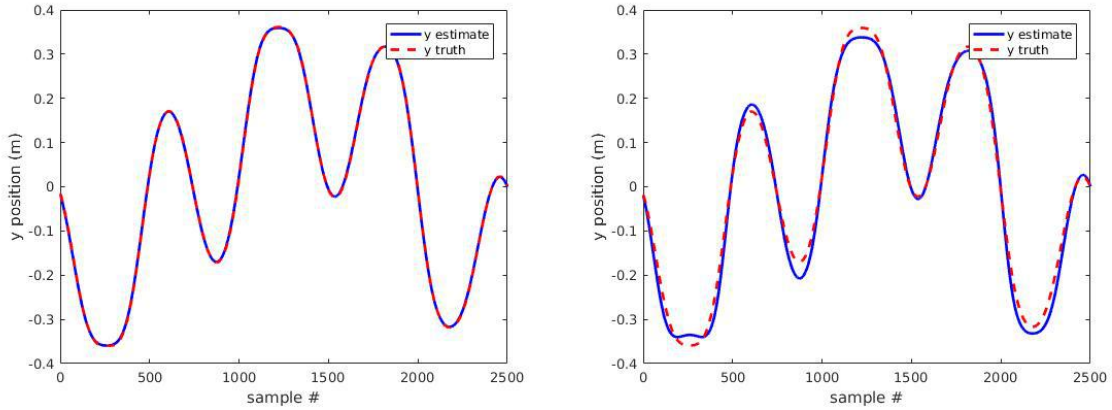


Figure 3.21: Left: Estimated data vs. ground truth for simulated y data using 2-D continuum joint model. Right: Estimated data vs. ground truth for simulated y data using a pin joint.

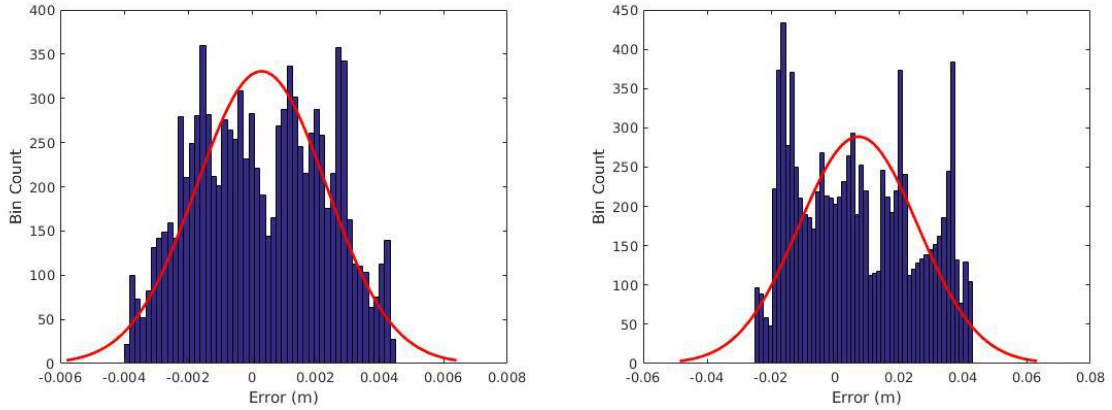


Figure 3.22: Left: Histogram of estimation error for simulated y data using 2-D continuum joint model. Right: Histogram of estimation error for simulated y data using pin joint model.

We can see from the histogram in Figure 3.22 that all the error is within ± 4 mm of error for the 2-D continuum model. Even though we trained on simulated data, error still exists because neural networks are function approximators. When comparing the results from the pin joint model to the 2-D continuum joint model the resulting worst case error is an order of magnitude greater, resulting in an error of ± 4 cm. These results verified the trainability of the neural network, but they also demonstrated how much error the pin joint assumption can introduce when applied to continuum joints. Since the difference in errors between the pin joint model and the continuum joint model are large, it would be better to use the continuum joint model for modeling rotation instead of the pin joint model as has been done in the past.

Pin Joint Model and 2-D continuum Joint

After verifying that the RNN architecture could learn the 2-D continuum joint model using simulated data, we wanted to see how accurate our estimations could be for the same models trained on data from the physical sensor. We again began by testing the pin joint model and compared the resulting *xyz* estimates to the continuum joint model estimates. Since we only estimate θ, shown in Figure 3.17 for the pin joint model, we can only estimate the end-effector position using the *y* and *z* position, and assume that the estimated *x* position is 0.

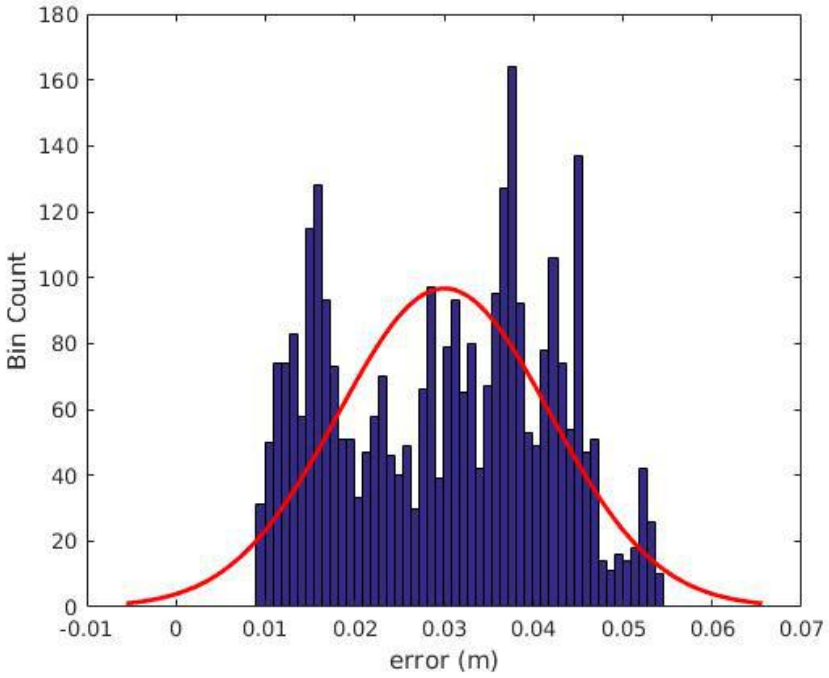


Figure 3.23: A histogram of Cartesian error for the pin joint based parameterization.

Figure 3.23 shows a histogram of Cartesian error for the *xyz* estimate from the pin joint model. The this error is due to the changing length of the joint as well as the changing location of the joint center when the grub link rotates. This requires us to use a higher fidelity model that can more accurately model the changing length of the joint as well as the changing joint center position.

Next, we tested training the 2D continuum joint model shown in Figure 3.19 on data from the physical sensor. For this model the arc length, δ, is not assumed to be constant. The hope is

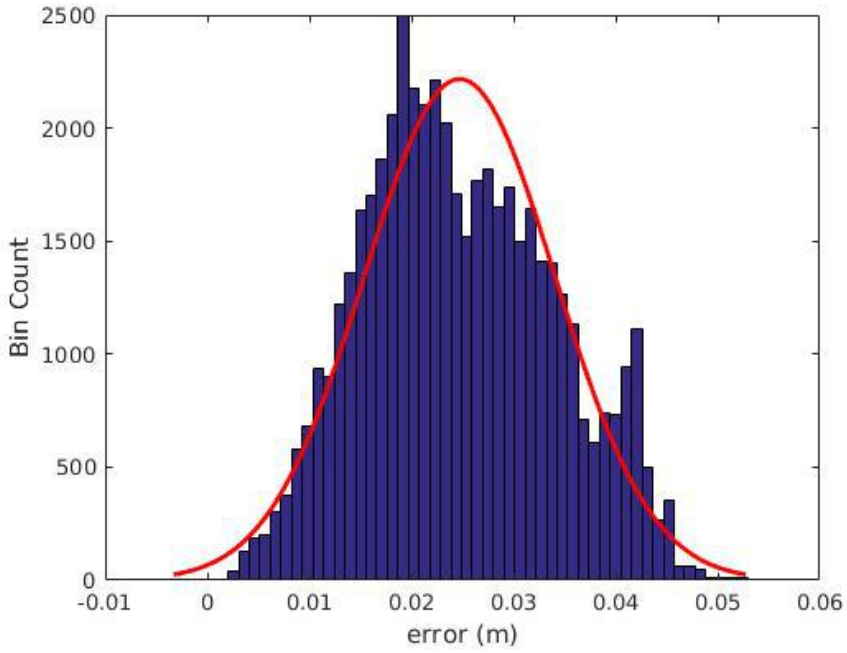


Figure 3.24: A histogram of Cartesian error for the 2-D continuum joint based parameterization.

that the neural network would learn how to estimate the arc length δ and the rotation angle θ. This would hopefully increase the accuracy in the y and z estimates of the end-effector position. Once again the estimate of the end-effector's x position was assumed to be 0. Figure 3.24 shows the resulting Cartesian error resulting from use of the 2-D continuum joint model. We can see that the *xyz* estimation accuracy of the 2-D continuum joint model was better than the *xyz* estimates from the pin joint model. This demonstrates that increasing model complexity in two dimensions can lead to a decrease in estimation error. Following these results, the next step was to determine the neural networks ability to estimate end-effector position in three dimensions.

3.8 3D Training Validation and Results

As was discussed previously in the analytical model section, due to defects inherent to manufacturing fabric-based robots by hand, there are many unpredictable ways that the grub can move when actuated. Unpredictable motion can also occur depending on which direction the grub's link is rotating. As the distal link rotates about the x-axis, the antagonistic bladders fill and vent at different rates and the joint lengthens and shrinks, resulting in hysteresis. This is especially

apparent when comparing the noisy, inconsistent raw ADC values from the string potentiometers in Figure 3.25 to the smooth consistent paths of the simulated ADC values in Figure 3.20.

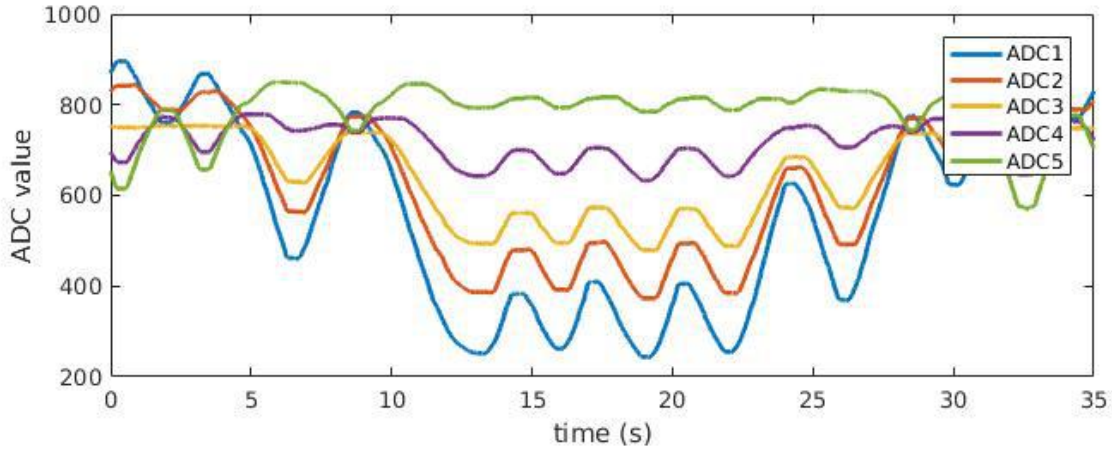


Figure 3.25: raw ADC values from one of the physical string potentiometer array.

Since we know that the neural network approach works on simulated data, our next task was to verify the estimation accuracy using the various parameterizations discussed in section 3.6 with data from the real platform.

3.8.1 Data Collection

Data collection for the machine learning portion consisted of two parts, collecting ground truth and collecting Analog to Digital values from the sensor. The HTC Vive was again used to collect truth data from the grub. The information was then published over ROS and collected for every time step. The Arduino Pro Mini was used to collect the ADC values from the sensor and were passed into a laptop and published over ROS as well. Both the ground truth data and the ADC values were appended to arrays and saved as pickle files for speed. This data was then loaded into a python script and used to train a TensorFlow graph. A time series seqeuence of ADC values is passed into the TensorFlow graph based on the various robot parameter models mentioned previously, and an xyz position estimate is passed out of the graph. The ground truth data corresponding to the ADC value sequence would then be used to determine the training loss of the neural network. Since only a single output is generated, it is only necessary to compare

the value of the ground truth estimate that corresponds to the last ADC values in the time series sequence for the loss calculation. This process is repeated using randomly selected sequences.

3.8.2 Parameter-based Learning

In this section we discuss obtaining xyz estimates from our neural network using models based in three dimensions. We trained an RNN with GRU cells using three different three dimensional models. The data collection consisted of $400,000$ data points collected at 400Hz, and divided into training and testing groups using an 80/20 split. After every few thousand training iterations, a section of the test data was extracted and passed through the RNN to validate estimation accuracy and to make sure overfitting was not occurring. The graphs shown in Figure 3.26 are histograms of the error between the *xyz*-position estimates and the *xyz*-position ground truth that resulted from passing the section of test data through the RNN for each of the three 3-D models used.

We initially estimate parameters u, v, and h, from the homogeneous transformation in Equation 3.1 described in section 3.5.1. The neural network would now output three parameters, u, v, and h, and uses these parameters to estimate *xyz* position using the homogeneous transformation from Equation 3.8. We see an immediate increase in accuracy compared to the two dimensional methods discussed previously if we compare the top graph in Figure 3.26 with the 2-D results from the graph shown in Figure 3.24. The error decreased because we no longer assumed that the x position estimate for the end-effector location was 0. We also saw a decrease in y and z position estimates.

Because of this increase in accuracy when we increased the model complexity, our next question was whether or not we would get better accuracy by not only estimating u, v, and h, but also estimating link lengths L_1 and L_2. These link lengths were initially measured by hand and we wondered if we would see an increase in accuracy if the neural network learned the link lengths instead of passing in hard coded values. After training the network using parameters u, v, h, L_1, and L_2, we compared the resulting estimates to the estimates from the previous model. When comparing the top two graphs in Figure 3.26, upon first inspection it seems that just learning u, v, and h was more effective. However, the results from the middle figure are less spread out, which means that the estimation was less noisy and more normally distributed.

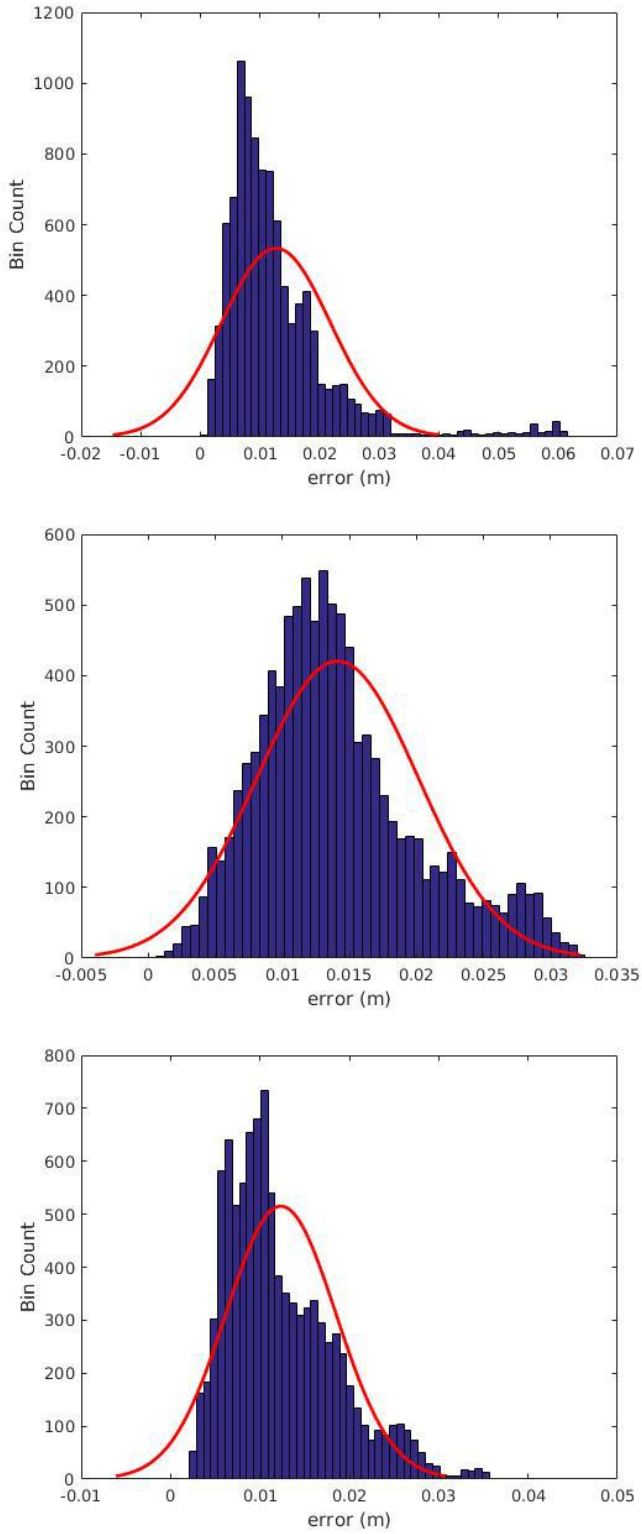


Figure 3.26: Top: Histogram of total *xyz* error estimating u, v, and h. Middle: Histogram of total *xyz* error estimating u, v, h, L_1, and L_2. Bottom: Histogram of total *xyz* error estimating all parameters.

The final step in verifying parameter-based learning was to estimate all of the parameters of the homogeneous transformations from Equation 3.8 including the x and y offsets found in $T_{O_2}^{O_1}$ and $T_{O_4}^{O_3}$. Ideally this approach would eliminate error that was potentially introduced by approximate measurements of xyz offsets.

Table 3.1: Comparison of RMS and median values for the three continuum joint models

GRU Cartesian Error			
Output Parameters	u,v,h	u,v,h,L_1,L_2	*all parameters*
Median	0.0104	0.0132	0.0108
RMS	0.0156	0.0154	0.0138

The bottom graph in Figure 3.26, showed that estimating u, v, h, the link lengths, and all potential offsets resulted in even greater accuracy. Upon reviewing all of the results from each of the 3-D models, we saw that learning all the parameters of the robot arm performed the best of all three methods. Table 3.1 shows the comparison of the RMS error and the median error of all three of these models.

3.8.3 Parameter-free Learning

Now that we have explored the effects of increasing the number of learned parameters on estimation accuracy, we need to determine whether learning parameters leads to better estimation than providing no model at all. For this scenario we learned x, y, and z directly. Our time series sequence was passed into the RNN and the output was the estimated end-effector position. Figure 3.27 shows a histogram of the resulting errors when parameter-free learning is used.

Table 3.2: Comparison of RMS and median values for the three continuum joint models compared to parameter-free learning.

GRU Cartesian Error				
Output Parameters	u,v,h	u,v,h,L_1,L_2	*all parameters*	*parameter-free*
Median	0.0104	0.0132	0.0108	0.0173
RMS	0.0156	0.0154	0.0138	0.0195

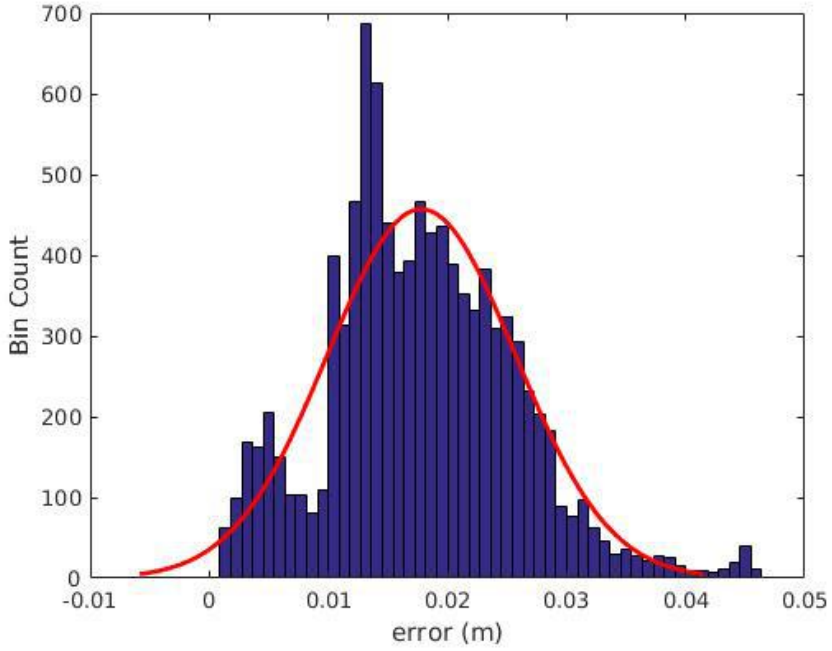


Figure 3.27: Histogram of estimation error for parameter-free learning.

Table 3.2 show the RMS error and the median error of the parameter-free method compared to the three continuum joint methods discussed previously. So far, the results from Table 3.2 indicate that parameter-based learning leads to better accuracy than parameter-free learning.

To increase understanding of the results from learning using these different models, we used a box and whisker plot, shown in Figure 3.28 to better understand the spread from the 5th percentile to the 95th percentile. Traditionally box and whisker plots quartiles 1 and 3 delineate the 25th and 75th percentiles respectively, but the code used to render the plots was modified to show the 5th and 95th percentiles instead. We expected the position estimation accuracy to increase as we increased the number of parameters used in our machine learning models.

3.8.4 Learning with Disturbances

All of the tests run thus far were done using the data discussed at the beginning of section 3.6, which was taken without any external disturbances. Supplemental video 3 shows how this data was obtained (https://youtu.be/LmUZaTansTc). One of the major research goals of this project was to determine if this type of sensor combined with machine learning would be able to estimate off-

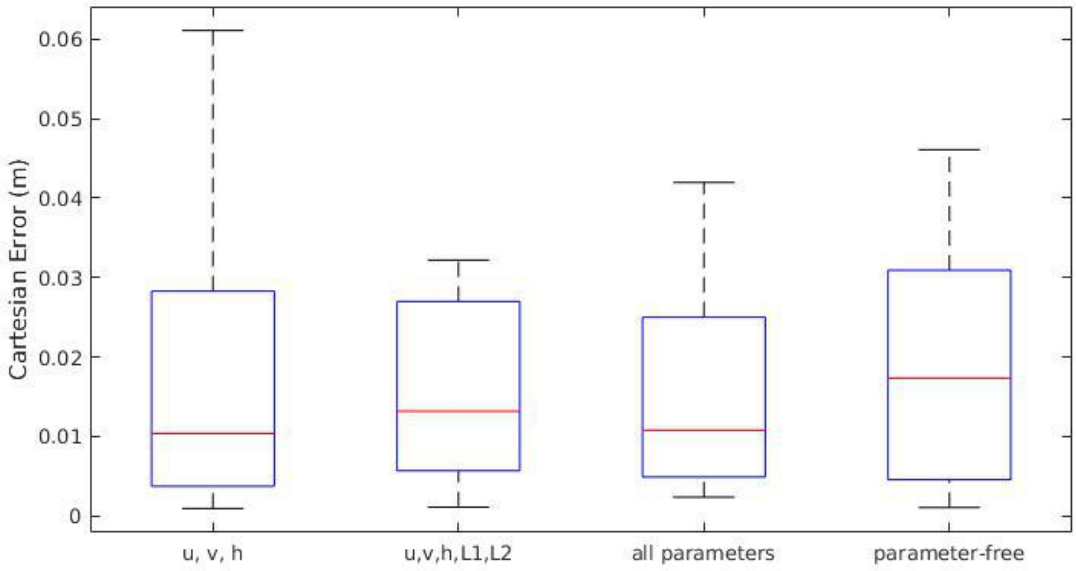


Figure 3.28: Box and whisker plot showing the Cartesian error of the three parameter-based methods along with the parameter-free method without any disturbances.

axis bending caused by disturbances and not just bending about the controlled axis of rotation. In order to test this, we recorded data in the same manner as before using pneumatic actuation, but this time we randomly perturbed the grub position by physically pushing it back and forth in the positive and negative x-directions as shown in supplemental video 4 (https://youtu.be/KR9x0nLDzws). The RNN was then trained again on this data. To verify the effectiveness of the RNN we compared the Euclidean distance error by taking the square root of the sum of the squares for the four scenarios displayed in Table 3.2. Table 3.3 shows the RMS error and median values of the estimation error for data without disturbances was used to train the RNN, along with the RMS and median values of the estimates from the RNN that was trained on data with disturbances included. Figure 3.29 shows a box and whisker plot that visualizes the spread of the Cartesian error in meters for each of the four methods. The results of this test did not turn out as expected. In our previous tests on the effects of parameter-based learning and parameter-free learning, we found that parameter-based learning had a lower estimation error than parameter-free learning. However, if we add disturbances, such as off-axis bending from pushing on the Grub in the x-axis, the model based methods appear to break down, which seems to indicate that our models were not accurate enough to account for disturbances. This shows that when more accurate models are used, parameter-based

Table 3.3: A comparison of RMS and median values for parameter-based and parameter free methods trained on data without disturbance and with disturbance.

GRU Euclidean Distance Error					
Scenario		*u*,*v*,*h*	*u*,*v*,*h*,L_1,L_2	*all parameters*	*parameter-free*
Without Disturbance	RMS	0.0156	0.0154	0.0138	0.0195
	Median	0.0104	0.0132	0.0108	0.0173
With Disturbance	RMS	0.0342	0.0454	0.0327	0.0223
	Median	0.0123	0.0358	0.0269	0.0174

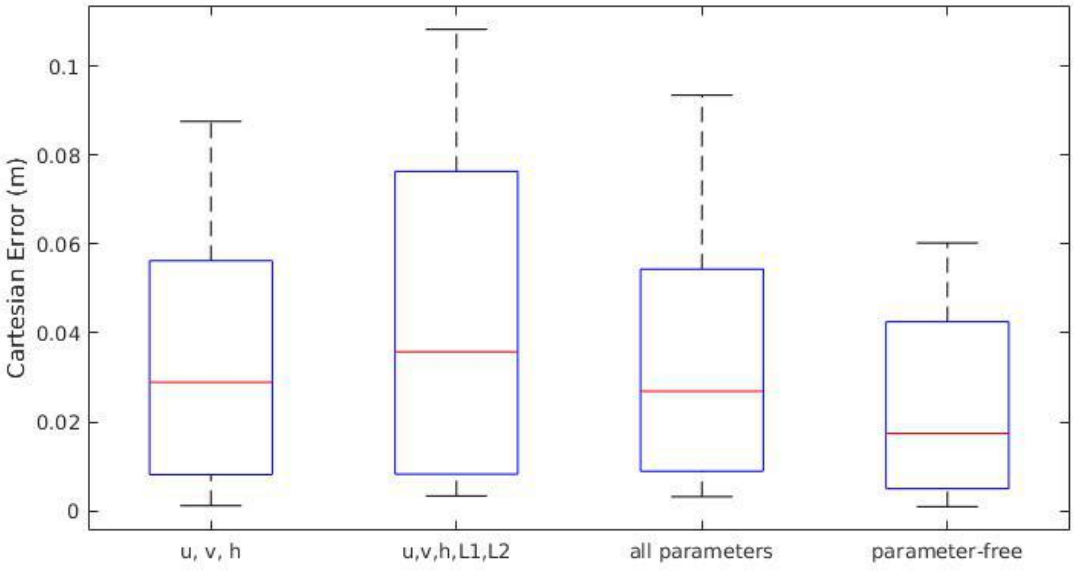


Figure 3.29: Box and whisker plot showing the Cartesian error of the three parameter based methods and the parameter free method for learning with disturbance.

methods have greater estimation accuracy than parameter-free methods. If the model is not a close enough representation of how the robot actually moves, or if a model is too difficult to develop, then the parameter-free method would yield better results, and be more robust to disturbances if the network is trained on data that includes disturbances.

3.8.5 Comparison of Neural Network Architectures

Gated Recurrent Units (GRU)

Thus far we have used an RNN network that employed GRU cells as their gated structure. Because it performed so well, we wanted to see if other types of neural network architectures could perform as well or better than using an RNN with GRU cells. The mathematical representation of a GRU is shown in Equation 3.10.

$$\begin{aligned} z_t &= \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \\ r_t &= \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \\ h_t &= (1 - z_t) \circ h_{t-1} + z_t \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h) \end{aligned} \tag{3.10}$$

x_t : input vector

h_t : output vector

z_t : update gate vector

r_t : reset gate vector

The $\circ$ operator is known as the Hadamard product, where two matrices are multiplied together element wise to create a third matrix. z_t determines how much of the previous information gets passed on to the output vector. r_t determines how much past information is forgotten and not passed on to the output vector. W, U and b are simply weight matrices and vectors, much like the weight matrices and vectors used in a fully-connected layer network.

Since learning all parameters performed better than learning a few parameters, and parameter-free learning also performed well, we only compare the estimates of these two methods for the various network architectures. The first architecture we explored was an RNN structure that was very similar to GRU's, called long short term memory cells, or LSTM's. The second architecture we explored was a convolutional neural network instead of an RNN.

Long Short Term Memory (LSTM)

LSTM's are very similar to GRU's. They have long and short term memory due to the state vector being passed directly through each successive layer. The only difference between GRU's and LSTM's is that LSTM's have one more gate, which can be seen in Equation 3.11.

$$\begin{aligned} f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\ i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\ h_t &= o_t \circ \sigma_h(c_t) \end{aligned} \tag{3.11}$$

x_t : LSTM input vector

f_t : forget gate's activation vector

i_t : input gate's activation vector

h_t : output gate's activation vector

c_t : cell state vector

In an attempt to determine which architecture is better for our purposes, we trained our neural network using LSTM cells instead of GRU cells on the data that was taken with disturbances added. Table 3.5 shows the results of the parameter-based all parameters method compared to the parameter-free method. Similar to GRU's, the parameter-free method has a more accurate median and RMS value. A comparison of LSTM's and GRU's is shown in Figure 3.32.

Table 3.4: A comparison of RMS and median values for parameter-based and parameter free methods using LSTM cells.

LSTM Euclidean Distance Error		
Output Parameters	*all parameters*	*parameter-free*
Median	0.0182	0.0136
RMS	0.0272	0.0196

For a more visual explanation Figure 3.30 shows the box and whisker plot for each the parameter-free learning method and the method that learned all the parameters.

The results from using LSTM cells are similar to the results from using the GRU cells, in that the parameter-free method performed better on disturbance data than the parameter-based method. The use of LSTM cells also resulted in better RMS and median values than the GRU cells, but suffered from a larger spread of error values as can be seen when comparing all of the results from the various architectures in Figure 3.32 to the all parameter and parameter-free boxplots.

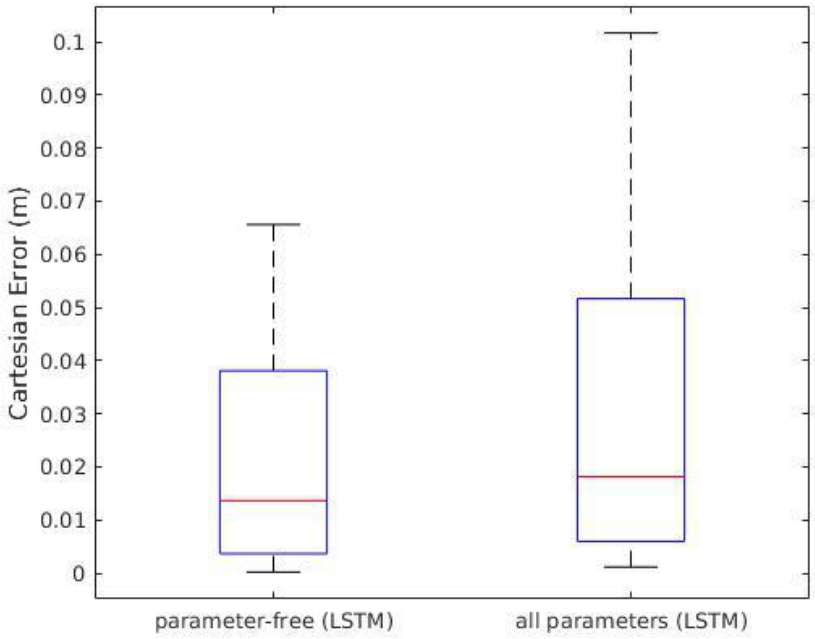


Figure 3.30: Box and whisker plot for parameter-based and parameter free methods when using LSTM cells.

Convolutional Neural Networks (CNN)

RNN's may not be the best architecture for learning this type of model, so we looked into using CNN's as a potential learning architecture. According to Ghering et al., RNN's were the go-to architecture for machine translation [34]. It was believed that RNN's performed better than CNN's in machine translation because of their memory. In [34], they found that the CNN was better at taking information and looking at it all in parallel instead of sequentially, thus the speed and accuracy of their results increased.

Table 3.5: A comparison of RMS and median values for parameter-based and parameter free methods using a CNN architecture.

CNN Euclidean Distance Error		
Output Parameters	*all parameters*	*parameter-free*
Median	0.0119	0.0170
RMS	0.0223	0.0209

A convolutional neural network is one where a filter of a specified size is convolved with the inputs of the neural network. For example, our input to the neural network may be a 10×100 matrix, where 10 is the number of ADC values and 100 represents a time series sequence of 10 ADC inputs. We would then convolve a convolutional kernel of size 3×10 with our 10×100 matrix of inputs, resulting in a filter of size 1×100. This would be repeated n times resulting in n number of filters. These filters would be stacked together to form an $n\times100$ matrix. The convolution can then be repeated any number of times. The final $n\times100$ matrix is then reshaped into a $1\times(100n)$ array and passed through a fully-connected layer to obtain the size of the desired output array. The box and whisker plot for this method is shown in Figure 3.31.

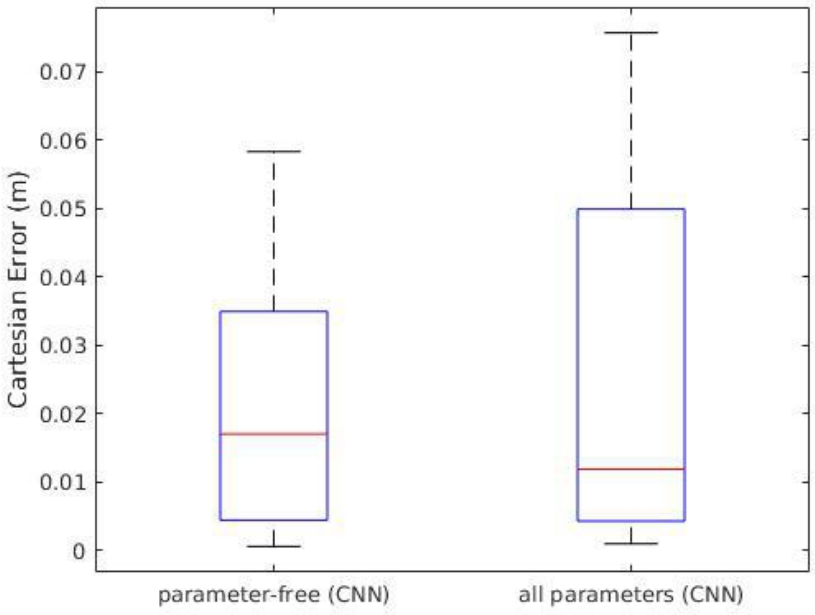


Figure 3.31: Box and whisker plot for parameter-based and parameter free methods when using a CNN.

Comparison of All Architectures

Now that these three architectures have been explored we needed to see how each of them compare to determine the best architecture for machine learning with this type of sensor. Figure 3.32 shows boxplots of all of the neural network architectures in the same plot for an easier comparison.

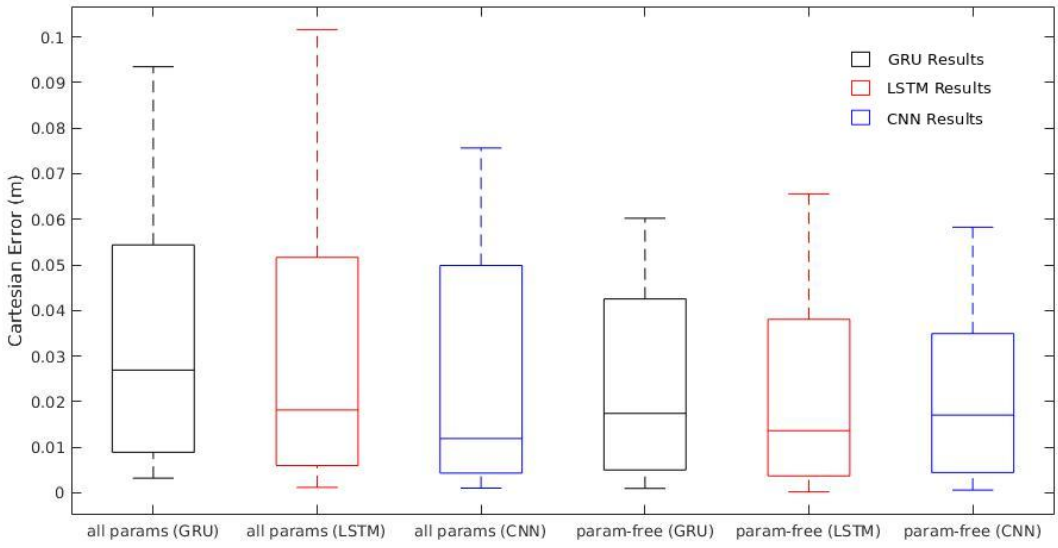


Figure 3.32: Box and whisker plot for parameter-based and parameter free methods comparing all three neural network architectures trained on disturbance data.

Based on these results we can say that the parameter-free methods performed better for each of the neural network architectures. At first glance it appears that the CNN performed better than the LSTM because the 95th percentile is lower and the spread is smaller as well. To determine which architecture performed best, we needed to look at a statistical analysis to better compare the averages as well as the spreads of the data sets. Using MATLAB's *ttest2* function, we compared the means of two sample sets at a time. Using the '*Tail*' option with the '*left*' argument, we were able to use an alternate hypothesis that determined whether the mean from the sample on the left was less than the mean of the sample on the right. The following is the list of sample pairs that we compared using the *ttest2* function:

1. GRU all params (GRU-AP) to CNN all params (CNN-AP)
2. GRU all params (GRU-AP) to LSTM all params (LSTM-AP)
3. LSTM all params (LSTM-AP) to CNN all params (CNN-AP)
4. GRU param-free (GRU-PF) to CNN param-free (CNN-PF)
5. GRU param-free (GRU-PF) to LSTM param-free (LSTM-PF)
6. LSTM param-free (LSTM-PF) to CNN param-free (CNN-PF)

Table 3.6 shows the resulting P-values and H-values of the sample pair comparisons. H represents whether to accept or reject the null hypothesis, with 1 representing "reject the null" and 0 representing "cannot reject the null". P represents with what certainty we can reject or accept the null hypothesis and ranges between 0 to 1.

Table 3.6: A comparison of the various architectures to determine statistical significance.

Statistical Comparison		
Sample Pairs	P-value	H-value
GRU-AP↔CNN-AP	1	0
GRU-AP↔LSTM-AP	1	0
LSTM-AP↔CNN-AP	1	0
GRU-PF↔CNN-PF	1	0
GRU-PF↔LSTM-PF	1	0
LSTM-PF↔CNN-PF	0	1

Based on these results we determined that the CNN-AP was the best architecture for the parameter based methods. We also determined that the GRU-AP method performed better than the LSTM-AP method. For the parameter free methods, LSTM-PF performed the best, while the CNN-PF outperformed the GRU-PF once again. The means for the LSTM-PF and CNN-PF respectively are 0.0174 m and 0.0209 m, which is a difference of 3.5 mm. Although the LSTM performed marginally better, the CNN trained faster and was able to use a longer time series sequence of data.

3.9 Testing Network on Live Data

All results thus far have been results taken from training and running test data through the neural network for validation. These test results showed that the best architecture to use for this type of estimation is a Convolutional Neural Network. Figure 3.33 shows plots of the *xyz* estimates compared to the *xyz* ground truth values. Long short term memory RNN's seemed to have better averages and median values than the GRU, but they also had a larger spread in error. To validate the results from training these various neural networks, we took the trained networks and passed realtime data through them at 400 Hz in time series sequences. Since most of our time was spent with GRU's we will compare the estimates from the GRU based networks to the estimates from the CNN based networks. To confirm the results shown in Figure 3.32, where the CNN estimates were more accurate than the GRU estimates, we tested the networks trained on the parameter-free models as well as the parameter-based models for the GRU and CNN architectures.

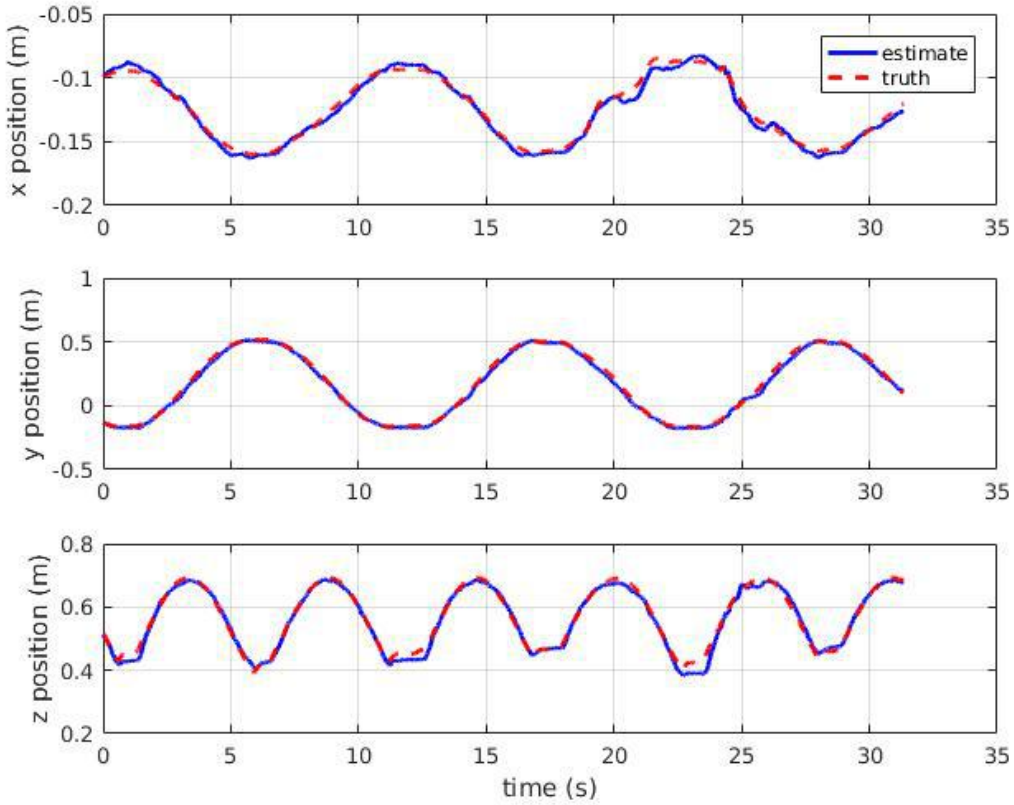


Figure 3.33: Plot showing the *xyz* estimates vs. *xyz* truth for a parameter-free CNN.

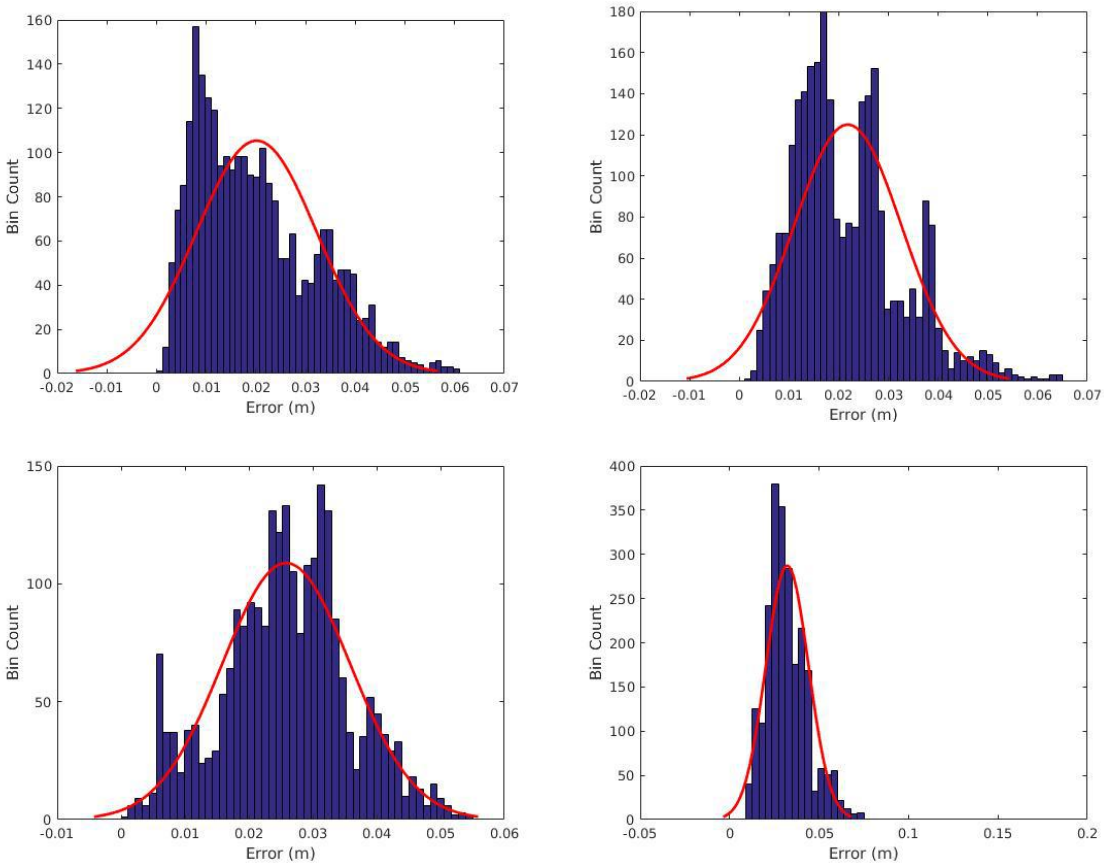


Figure 3.34: Top Left: Histogram of error for live testing of the parameter free CNN. Top Right: Histogram of error for live testing of all parameters CNN. Bottom Left: Histogram of error for live testing of parameter free GRU. Bottom Right: Histogram of error for live testing of all parameters GRU.

We did live testing on four different networks: 1) Parameter-based CNN, 2) Parameter-free CNN, 3) Parameter-based GRU, and 4) Parameter-free GRU. In order to visualize the spread of the error, we generated four histograms for the Cartesian error (see Figure 3.34). We can see from the top left and right Figure in Figure 3.34 that the CNN estimation error is closer to zero than the errors generated from the GRU architectures.

Figure 3.32 provides a boxplot that allows us to see what values are contained within the 95th percentile. The parameter-free CNN performed the best of all four networks. Although the parameter-free GRU network had a smaller spread, its median value is still higher than either of the CNN networks.

Since the rate at which sensor feedback occurs is important to control methods, we tracked the estimation rates for each. The estimation rate for the CNN was about 80 Hz, while the estimation rate for the GRU was around 70 Hz. This is especially impressive, because the sequence length used for the GRU was 75 while the sequence length for the CNN was 200. This means that the CNN was looking at data for a time window of 0.5 seconds, while the GRU was looking at a time window of 0.1875 seconds. This shows that the CNN is much faster than the RNN architecture and can process a longer time series sequence of data. Even though the parameter-free LSTM performed marginally better than the parameter-free CNN, CNN's can be trained faster and run faster, which is one reason to choose CNN's over LSTM architectures.

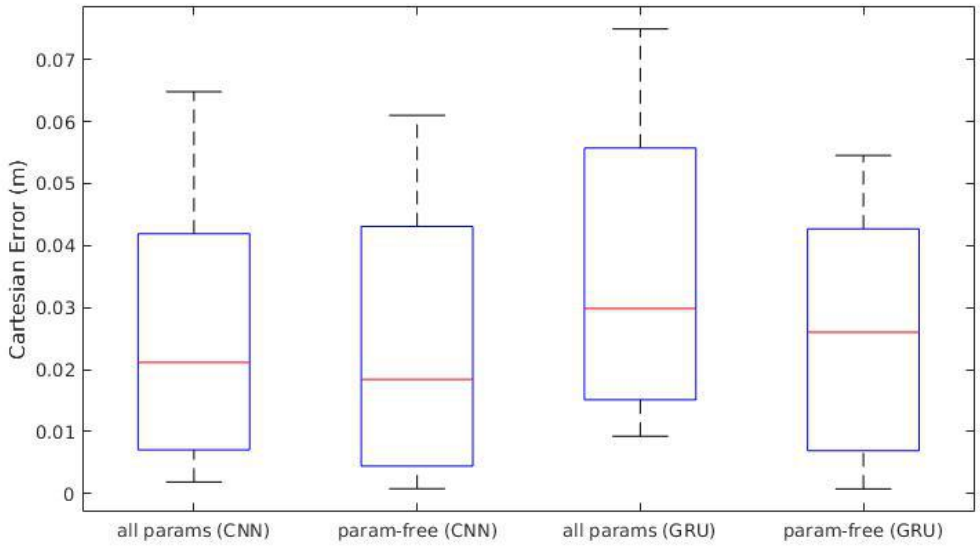


Figure 3.35: Box and whisker plot for parameter-based and parameter-free methods for the CNN and GRU network architectures.

CHAPTER 4. CONCLUSION

4.1 Future Work for Soft-robot Tactile Sensing

Calibrating the tactile sensor was not within the scope of this work. Although a simple calibration was used for preliminary testing, more work can be done on developing a robust calibration that can account for the hysteresis of the fabric based structure. We only focused on calibrating the tactile sensor when the taxels were undergoing loading. To use this type of sensor for control methods, a calibration will need to be developed that will also work for the case when the taxels are being unloaded.

4.1.1 Contributions in Soft-robot Tactile Sensing

This thesis has presented robust tactile sensor arrays that can measure contact over large areas, which is something that is difficult for most other current designs. Additionally, the sensor design is flexible and scalable for manufacturing, which makes it an ideal candidate for integration for future applications with soft robots. We have presented two methods for overcoming cross-talk. The effects of parallel parasitic cross-talk can be modeled and used to calculate the actual resistance at each taxel and correlated to force, or cross-talk can be eliminated through additional circuitry, i.e. multiplexers and diodes, so that voltage output can be directly correlated to force. The first method allows sensors to be built with minimal circuitry at low cost, and the second method simplifies the typical switching method in the literature by using diodes to stop the negative flow of current through the circuit. Both methods are effective at showing contact locations and estimating forces applied to the tactile sensor. We expect that these results, coupled with past research on whole-body tactile sensing for control in cluttered environments (see [35, 36]), should enable soft robots to more effectively interact with the real world. Additionally, the low cost of having such a

large number of sensors enabled by our methods may be useful for learning data-driven models of soft robots and soft robot interaction, given the inherent difficulty in modeling them analytically.

4.2 Future Work for Soft-robot End-effector Position Estimation

There is still much work that can be done with machine learning soft-robotic motion since this field is relatively new. Further questions remain for how to develop more effective training strategies, how to better integrate this type of sensor into the structure of the soft-robot limb, and how to improve the sensor design. For more effective training strategies, research can be done to determine how to collect data more efficiently to result in better training sets. Work can also be done on the structure of the CNN and RNN networks to determine how parameters, such as number of layers or length of state vector, should be configured for increased estimation accuracy. Improved sensor design could also help increase the estimation capability of the neural network. More research can be done to determine how to place the string potentiometers to better capture the motion of the soft-robot limb. This type of sensor could also be moved to the interior of the robot structure to protect it from contact from outside objects that could affect the estimation readings. This, combined with improved learning methods, could significantly reduce the amount of error currently in the system and lead to improved robot position estimates.

4.2.1 Contributions in Soft-robot End-effector Position Estimation

Position estimation methods for soft robot applications have also been presented in this thesis. A method for estimating motion in multiple degrees of freedom using redundant string potentiometers was developed and characterized successfully through machine learning. Multiple neural network architectures were discussed, and it was determined that although Recursive Neural Networks are an effective architecture for characterizing systems with hysteresis, Convolutional Neural Networks performed more accurately and efficiently. Two methods of learning using RNN's and CNN's were explored, namely parameter-free and parameter-based methods. We found that learning parameters was more accurate when the data used for training was free of disturbances. When data with disturbances was used, the parameter-free method performed better. This is likely due to the resulting motion not lining up with the model that we used to approximate

the motion of the soft robot limb. If the model parameters used for characterizing the motion of the robot is close to the actual motion, even with the presence of noise and hysteresis, parameter-based learning would be a more accurate way to do machine learning. If a motion model cannot be developed due to the complexity of the robot motion, or if a motion model is not a close enough approximation, parameter-free machine learning is a more effective method for machine learning.

REFERENCES

[1] Rogers, J. A., Someya, T., and Huang, Y., 2010. "Materials and mechanics for stretchable electronics." *Science,* **327**(5973), pp. 1603–1607.

[2] Lu, N., Lu, C., Yang, S., and Rogers, J., 2012. "Highly sensitive skin-mountable strain gauges based entirely on elastomers." *Advanced Functional Materials,* **22**(19), pp. 4044–4050.

[3] Someya, T., Kato, Y., Sekitani, T., Iba, S., Noguchi, Y., Murase, Y., Kawaguchi, H., and Sakurai, T., 2005. "Conformable, flexible, large-area networks of pressure and thermal sensors with organic transistor active matrixes." *Proceedings of the National Academy of Sciences of the United States of America,* **102**(35), pp. 12321–12325.

[4] Someya, T., Sekitani, T., Iba, S., Kato, Y., Kawaguchi, H., and Sakurai, T., 2004. "A large-area, flexible pressure sensor matrix with organic field-effect transistors for artificial skin applications." *Proceedings of the National Academy of Sciences of the United States of America,* **101**(27), pp. 9966–9970.

[5] Yamaguchi, A., and Atkeson, C. G., 2016. "Combining finger vision and optical tactile sensing: Reducing and handling errors while cutting vegetables." In *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*, IEEE, pp. 1045–1051.

[6] Sareh, S., Jiang, A., Faragasso, A., Noh, Y., Nanayakkara, T., Dasgupta, P., Seneviratne, L. D., Wurdemann, H. A., and Althoefer, K., 2014. "Bio-inspired tactile sensor sleeve for surgical soft manipulators." In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, IEEE, pp. 1454–1459.

[7] Canavese, G., Stassi, S., Fallauto, C., Corbellini, S., Cauda, V., Camarchia, V., Pirola, M., and Pirri, C. F., 2014. "Piezoresistive flexible composite for robotic tactile applications." *Sensors and Actuators, A: Physical,* **208**, pp. 1–9.

[8] Yang, Y.-J., Cheng, M.-Y., Shih, S.-C., Huang, X.-H., Tsao, C.-M., Chang, F.-Y., and Fan, K.-C., 2010. "A 32x32 temperature and tactile sensing array using PI-copper films." *The International Journal of Advanced Manufacturing Technology,* **46**(9-12), feb, pp. 945–956.

[9] Yang, Y.-J., Cheng, M.-Y., Chang, W.-Y., Tsao, L.-C., Yang, S.-A., Shih, W.-P., Chang, F.-Y., Chang, S.-H., and Fan, K.-C., 2008. "An integrated flexible temperature and tactile sensing array using pi-copper films." *Sensors and Actuators A: Physical,* **143**(1), pp. 143–153.

[10] Wu, J., and Wang, L., 2016. "Cable Crosstalk Suppression in Resistive Sensor Array with 2-Wire S-NSDE-EP Method." *Journal of Sensors,* **2016**, pp. 1–9.

[11] Shimojo, M., Namiki, A., Ishikawa, M., Makino, R., and Mabuchi, K., 2004. "A tactile sensor sheet using pressure conductive rubber with electrical-wires stitched method." *IEEE Sensors Journal,* **4**(5), pp. 589–596.

[12] Zhang, T., Liu, H., Jiang, L., Fan, S., and Yang, J., 2013. "Development of a Flexible 3-D Tactile Sensor System for Anthropomorphic Artificial Hand." *IEEE Sensors Journal,* **13**(2), feb, pp. 510–518.

[13] Bhattacharjee, T., Jain, A., Vaish, S., Killpack, M. D., and Kemp, C. C., 2013. "Tactile sensing over articulated joints with stretchable sensors." In *2013 World Haptics Conference (WHC)*, IEEE, pp. 103–108.

[14] Marchese, A. D., Komorowski, K., Onal, C. D., and Rus, D., 2014. "Design and control of a soft and continuously deformable 2D robotic manipulation system." In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 2189–2196.

[15] Gerboni, G., Diodato, A., Ciuti, G., Cianchetti, M., and Menciassi, A., 2017. "Feedback Control of Soft Robot Actuators via Commercial Flex Bend Sensors." *IEEE/ASME Transactions on Mechatronics,* **22**(4), pp. 1881–1888.

[16] Yuen, M. C., Tonoyan, H., White, E. L., Telleria, M., and Kramer, R. K., 2017. "Fabric sensory sleeves for soft robot state estimation." In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 5511–5518.

[17] She, Y., Li, C., Cleary, J., and Su, H.-J., 2015. "Design and Fabrication of a Soft Robotic Hand With Embedded Actuators and Sensors." *Journal of Mechanisms and Robotics,* **7**(2), may, p. 021007.

[18] Elgeneidy, K., Lohse, N., and Jackson, M., 2016. "Data-Driven Bending Angle Prediction of Soft Pneumatic Actuators with Embedded Flex Sensors." *IFAC-PapersOnLine,* **49**(21), pp. 513–520.

[19] Gibbs, P. T., and Asada, H. H., 2005. "Wearable Conductive Fiber Sensors for Multi-Axis Human Joint Angle Measurements." *Journal of NeuroEngineering and Rehabilitation,* **2**(1), p. 7.

[20] Dobrzynski, M. K., Pericet-Camara, R., and Floreano, D., 2011. "Contactless deflection sensor for soft robots." In *IEEE International Conference on Intelligent Robots and Systems*, IEEE, pp. 1913–1918.

[21] Felt, W., Suen, M., and Remy, C. D., 2016. "Sensing the motion of bellows through changes in mutual inductance." In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 5252–5257.

[22] Yuan, Y., Wu, G., Li, X., Fan, Y., and Wu, X., 2011. "Effects of twisting and bending on LP_21 mode propagation in optical fiber." *Optics Letters,* **36**(21), p. 4248.

[23] You, X., Zhang, Y., Chen, X., Liu, X., Wang, Z., Jiang, H., and Chen, X., 2017. "Model-free control for soft manipulators based on reinforcement learning." In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 2909–2915.

[24] Melingui, A., Escande, C., Benoudjit, N., Merzouki, R., and Mbede, J. B., 2014. "Qualitative approach for forward kinematic modeling of a Compact Bionic Handling Assistant trunk." *IFAC Proceedings Volumes (IFAC-PapersOnline),* **19**, pp. 9353–9358.

[25] Merzouki, R., Melingui, A., and Mbede, J., 2014. "Compact bionic handling arm control using neural networks." *Electronics Letters,* **50**(14), pp. 979–981.

[26] Kerpa, O., Weiss, K., and Worn, H. "Development of a flexible tactile sensor system for a humanoid robot." In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, Vol. 1, IEEE, pp. 1–6.

[27] Russell, R., 1987. "Compliant-skin tactile sensor." In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, Vol. 4, IEEE, pp. 1645–1648.

[28] Strohmayr, M., Saal, H. P., Potdar, A., and Van Der Smagt, P., 2010. "The dlr touch sensor i: A flexible tactile sensor for robotic hands based on a crossed-wire approach." In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, IEEE, pp. 897–903.

[29] Day, N., Penaloza, J., Santos, V. J., and Killpack, M. D., 2018. "Scalable fabric tactile sensor arrays for soft bodies." *Journal of Micromechanics and Microengineering,* **28**(6), p. 064004.

[30] Horii, T., Nagai, Y., Natale, L., Giovannini, F., Metta, G., and Asada, M., 2014. "Compensation for tactile hysteresis using gaussian process with sensory markov property." In *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, IEEE, pp. 993–998.

[31] TEXAS INSTRUMENTS, 1998. *High-Speed CMOS Logic 16-Channel Analog Multiplexer/Demultiplexer.*, Feb. Revised July 2003.

[32] Bodily, D. M., 2017. "Design optimization and motion planning for pneumatically-actuated manipulators.".

[33] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems Software available from tensorflow.org.

[34] Gehring, J., Auli, M., Grangier, D., and Dauphin, Y. N., 2016. "A convolutional encoder model for neural machine translation." *arXiv preprint arXiv:1611.02344*.

[35] Jain, A., Killpack, M. D., Edsinger, A., and Kemp, C. C., 2013. "Reaching in clutter with whole-arm tactile sensing." *The International Journal of Robotics Research,* **32**(4), pp. 458–482.

[36] Killpack, M. D., Kapusta, A., and Kemp, C. C., 2016. "Model predictive control for fast reaching in clutter." *Autonomous Robots,* **40**(3), pp. 537–560.