Using Tactile Feedback and Gaussian Process Regression in a Dynamic System to Learn New Motions

MCE 499H Honors Thesis

Cleveland State University
Washkewicz College of Engineering
Department of Mechanical Engineering
May 13, 2016

Submitted by: Titus Lungu
Thesis Advisor: Eric Schearer, PhD

Abstract

This paper presents two proofs of concept which use Gaussian Process Regression and/or tactile feedback to learn how to control the motion of a dynamic system. The first proof of concept is presented in the form of a computer animation of a pendulum. The second is a robotic pendulum to which tactile feedback is delivered to teach it new motions outside of the training data set. Both proofs are designed in order to test the hypotheses of whether GPR can be used to learn how to control the motion of a mechanism and if tactile feedback can be given to said mechanism to show it a new motion to reproduce. This research is conducted as a prerequisite to the overall goal of implementing a method for a caregiver to deliver tactile feedback to a person with an arm neuroprosthesis in order to teach them new motions or improve a previously executed motion.

1. Introduction

Functional Electrical Stimulation (FES) is a method used to reanimate the arms or legs of people with paralysis (Smith et al., 1987). One implementation of this technology is an FES neuroprosthesis used for restoring functionality to the arms of individuals with upper spinal cord injuries (Memberg et al., 2014; Ho et al., 2014). Such a neuroprosthesis consists of surgically implanted electrodes in a person's arm, which are connected to and stimulate paralyzed muscles based on commands from a connected control unit (Smith et al., 1987).

These commands can be generated in two ways. One way is to send the explicitly defined stimulations to each muscle at a specified time (Smith et al., 1987). This method of stimulating the arm can be hard to implement as small changes in stimulation can cause significantly differing motions of the arm. Muscles also change over time in both size and strength and therefore a stimulation that causes a certain action one day may cause a different one a few days later. Various starting positions and orientations of the arm may also cause varying reactions to a stimulation.

The other method of generating commands is having a control policy that uses a dynamic model of how the arm behaves to infer what stimulations are required to produce a certain motion (Schearer et al., 2014). This allows more flexibility in the commands that can be generated and gives the neuroprosthesis a degree of autonomy. It allows the control unit to predict what stimulations may be needed due to the expected behavior of the arm.

However, the complexity of the human arm makes it difficult to create a global model which takes all the contributing variables into account when predicting required stimulations. While these latent variables can prove difficult to quantify, there are two aspects of any system which are more readily available, namely the inputs and outputs of the system. While the specific functions that convert the inputs to outputs may not be known, a data set of these inputs and outputs can be used to infer the relationship between the two. This produces a direct mapping between the inputs and outputs, which in the case of FES are the stimulation commands and the motions produced, respectively. This can be done using machine learning, which attempts to learn the model of a system by relating a set of input data to the corresponding outputs.

For such a complex system as the human arm, it is also desirable to have a way for caregivers and family members to be able to intuitively provide feedback to the neuroprosthesis in a non-lab setting. This will allow the neuroprosthesis to improve its control of the wearer's arm without requiring the intervention of a technical specialist in the field. Kinesthetic teaching is a common way humans teach other humans a motion, guiding the learner through said motion, and is widely used in robotics as well (Akgun et al., 2012). For example, a parent teaching their child to write may hold the child's hand with the pencil and tactilely guide them through the act of writing. Placing force sensors on the wrist of the arm with the neuroprosthesis would allow a caregiver to "show" the neuroprosthesis how to move by guiding the arm, via the wrist, through the desired motion. The force sensor could then send the data from the caregiver's push to the controller, which in turn, would update the control policy used for stimulation.

This paper discusses two hypotheses related to the overall goal of neuroprosthesis feedback control: 1) Gaussian Process Regression (GPR), a form of machine learning, can be used to learn how to control the motion of a mechanism and 2) tactile feedback can be given to said mechanism to show it a new motion to reproduce. In order to test these hypotheses to either support or refute them, two proof of concepts were designed. The first proof is a computer simulation of a rotating pendulum that learns how to control itself. The second proof is a robotic pendulum built to first

learn how to control itself and then interpret force feedback, given by a user, to perform new motions.

2. Proof of Concept 1 - Simulation

The first proof of concept was used to test the first hypothesis in order to determine whether Gaussian Process Regression (GPR) can be used to learn how to control the motion of a mechanism. This involved creating a MATLAB simulation of a pendulum driven by a motor. GPR was used to create the mapping between the system inputs and outputs, as described in the Appendix. The torque of the motor was used as the output in the GPR model and the angular position and velocity of the pendulum (both with random added noise) were used as the inputs. Figure 1 shows the interface of the simulation, with the pendulum on the left and the position trajectory on the right.

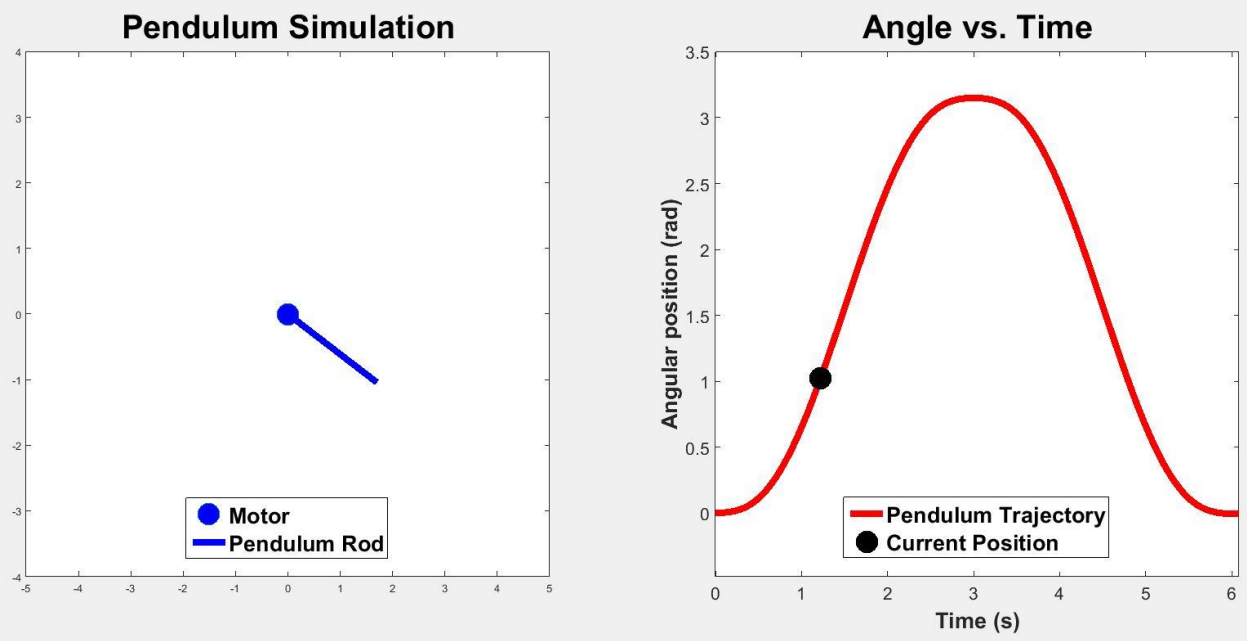

Figure 1: *Shown here is the interface of the computer simulation of the pendulum. The pendulum on the left oscillates about the motor. The pendulum trajectory is shown on the right in red, with the pendulum's current angular position indicated by the black marker. Both parts of the simulation update in real-time.*

The training data for training the GPR model consisted of the torques, positions, and velocities of the pendulum as it was swung from zero to pi radians. In order to create a smooth training data set, a system of equations was used to create smooth motions (Sciavicco and Siciliano, 2012). A five degree polynomial was used for the position curve, a four degree polynomial for the velocity curve, and a three degree polynomial for the acceleration curve

$$at^5 + bt^4 + ct^3 + dt^2 + et + x_{initial} = x_{final},$$
$$5at^4 + 4bt^3 + 3ct^2 + 2dt + e + \dot{x}_{initial} = \dot{x}_{final},$$
$$20at^3 + 12bt^2 + 6ct + 2d + \ddot{x}_{initial} = \ddot{x}_{final}.$$

Using the following six boundary conditions, the coefficients of the above polynomials are found, thus creating smooth position, velocity, and acceleration profiles for the system.

$$x_{initial} = 0,$$
$$x_{final} = \pi,$$
$$\dot{x}_{initial} = 0,$$
$$\dot{x}_{final} = 0,$$
$$\ddot{x}_{initial} = 0,$$
$$\ddot{x}_{final} = 0.$$

In order to produce the outputs for the training data which the GPR algorithm will use to learn a system model, the following equation was used to derive the torque of the pendulum, where *m* is the mass of the end of the pendulum and *L* is the length of the pendulum

$$T = mL^2\ddot{x}.$$

The learned GPR model was then used to predict the torques required to take the pendulum through a new motion, back from pi to zero radians, with equal and opposite velocity from the training data. The resulting motion of the pendulum was recorded and plotted against the desired motion to indicate the accuracy of the torque predictions, as shown in the *Results* section.

3. Proof of Concept 2 - Robotic Pendulum

The second proof of concept, a one degree-of-freedom robotic pendulum, was used to test both hypotheses of whether GPR can be used to learn how to control the motion of a mechanism and if tactile feedback can be given to said mechanism to show it a new motion to reproduce. A geared, brushless DC motor from Midwest Motion Products was used as the driving mechanism, with a 3-D printed pendulum link attached to the motor shaft and an optical encoder for acquiring position readings. The motor has a maximum torque of 99 in-lb. and a maximum speed of 124 RPM. The motor is controlled by a servo drive, also from Midwest Motion Products, with a continuous output current of 8 amps and which can receive a command voltage ranging from -10 to +10 volts. The command voltage is delivered from a National Instruments Elvis II+ prototyping board, which also receives the data from the encoder and force sensor. Finally, a resistance force sensor is attached to the pendulum link of the motor, allowing a user to deliver force feedback by pushing the force sensor to move the pendulum link in the desired fashion. The force sensor is only attached to one side of the pendulum link, therefore feedback may only be given to move the pendulum in one direction (counterclockwise, in the case of this setup). The entire system is shown in Figure 2.

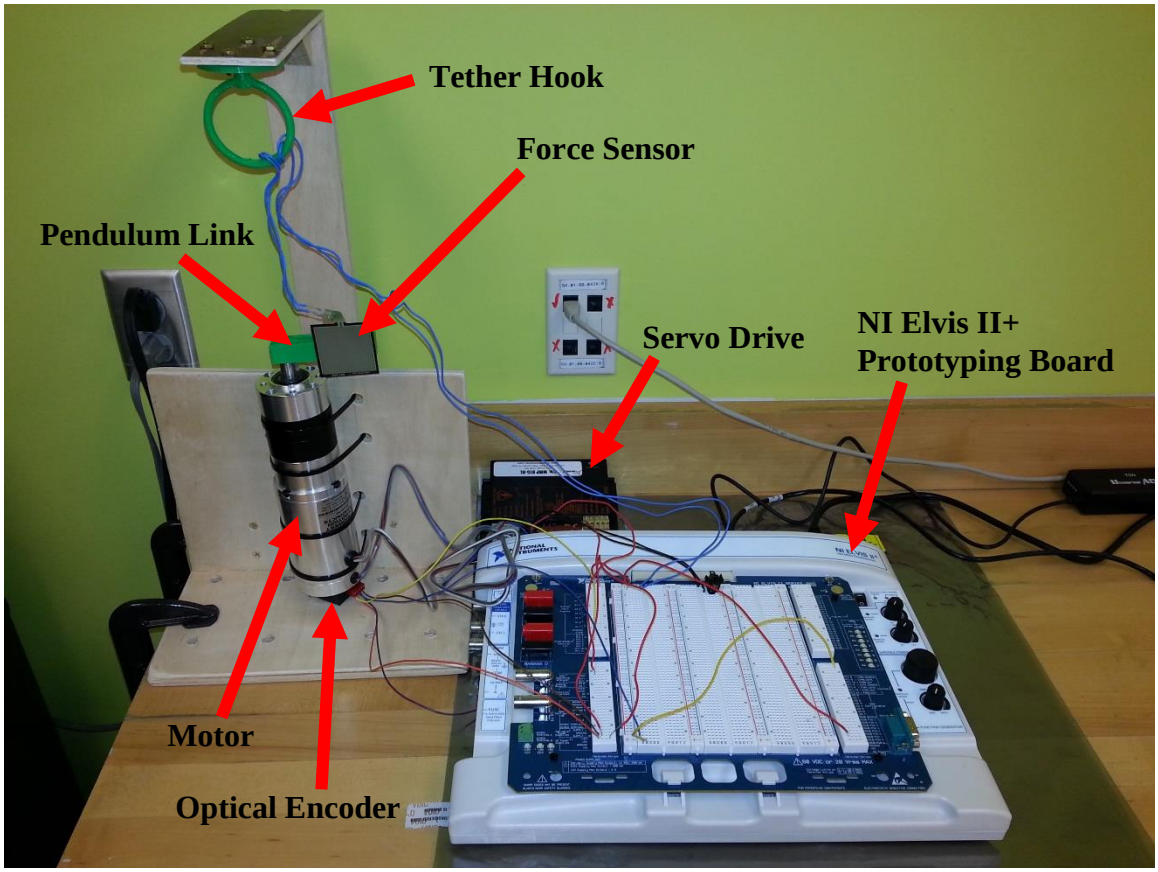

Figure 2: *Overview of the setup for the robotic pendulum. All the major components are labelled. Tactile feedback is delivered by pushing the green pendulum link at the force sensor. A force sensor is only placed on one side of the pendulum link such that it can only be pushed counterclockwise for delivering feedback.*

MATLAB's Data Acquisition Toolbox is used to gather data and deliver commands through the prototyping board. Position readings from a 1024 CPR quadrature encoder on the

motor are gathered at a frequency of 500 Hz and the command voltage is delivered to the servo drive at the same rate. Force sensor readings are generated by measuring the voltage drop across the force sensitive resistor, also at 500 Hz.

A mapping between the voltage commanded to the motor and the motion of the pendulum (angular position, velocity, and acceleration) was first learned to create a control policy that would allow the pendulum to perform new motions based on knowledge learned from the training data. The training data consisted of pairs of inputs and outputs to the system, which were voltages commanded to the servo drive and the pendulum motion produced, respectively. Command voltages are treated as outputs of the GPR model and the positions, velocities, and accelerations of the motor shaft are inputs. This way, the GPR model can be queried for some desired motion in order to predict the required voltages to produce that motion. The position is, of course, recorded via the optical encoder. This position data is then differentiated and the resulting noisy velocity and acceleration is filtered using a Butterworth filter (Figure 3).

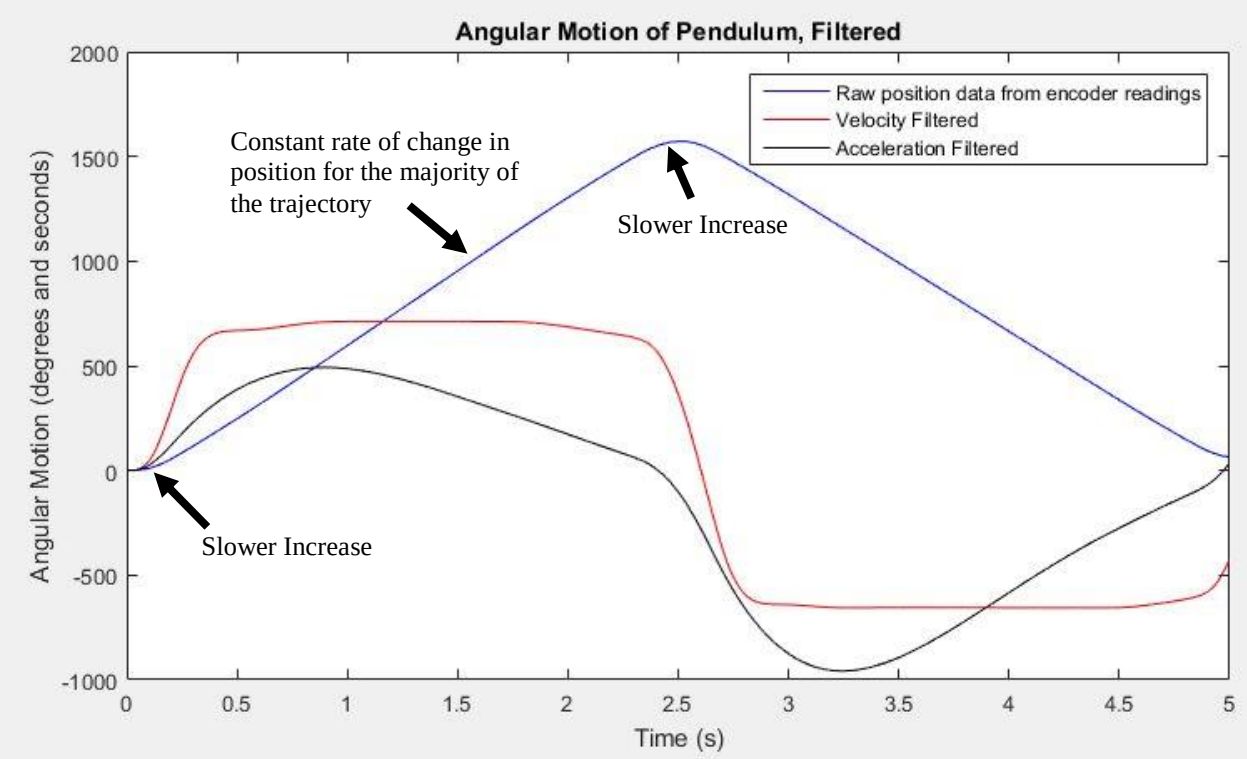

Figure 3: *Shown is an example of the angular motion of the pendulum, with position shown in blue. The pendulum can be described to oscillate in "periods". A period consists of the pendulum starting to move in a certain direction, followed by a constant motion in that direction, and finally a slowing down of movement in that direction. In this figure, a period occurs from 0 to about 2.5 seconds, and a second period follows from about 2.5 seconds to 5 seconds. Periods can be any size, depending on the amount of time for which they occur. As labeled, the position of the pendulum mostly changes at a constant rate during a period of motion, except for a short time of slower change at the beginning and end of the period.*

A second mapping was attempted to relate force feedback to the motion of the pendulum. This way, when feedback was delivered, the control algorithm could predict the desired motion, and then use that motion to predict the required voltage to give to the motor. However, the available force sensors had a low maximum force that could be applied and a long settling time after the force was removed, not allowing an accurate GPR mapping to be produced.

Due to this shortcoming in the available force sensors, it was assumed that the pendulum link is pushed at a constant average rate. Since any push on the force sensor that would result in the pendulum turning maxes out the force sensor, it can be said that for every time at which the force sensor reading is at the maximum value, the pendulum is rotating at said average rate. The value that implies a push has occurred is between 4.90 and 4.95 volts. Sometimes when the sensor is released and begins to settle, the voltage will overshoot this value. Therefore, it is assumed that a push has occurred only when the voltage value is within the specified range. If the force sensor reading is any other value, a push did not occur and the pendulum is not rotating. The voltage reading from the force sensor is very noisy and is therefore filtered with a Butterworth filter before it is used to predict when a push occurred. The output of the force sensor is depicted in Figure 4, with three labeled locations where a push occurred.

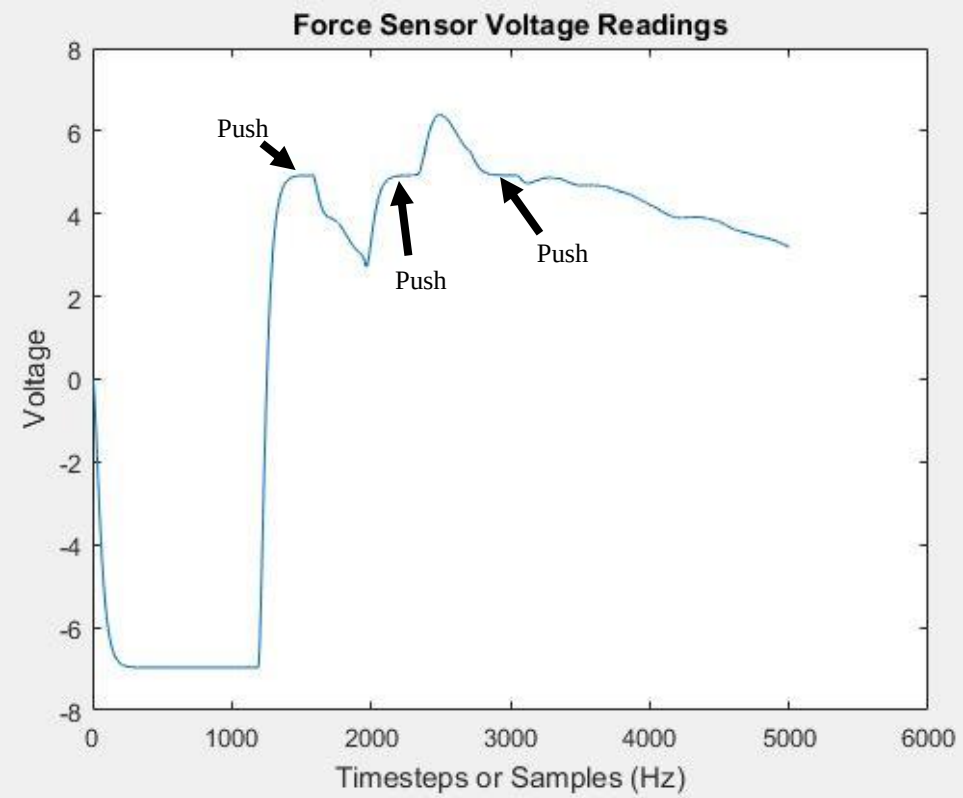

Figure 4: *After Butterworth filtering, the reading of the force sensor after tactile feedback is delivered may look like this. A push is said to have occurred for all places where the sensor reading is between 4.90 and 4.95 volts, as labeled in this example.*

When the pendulum is pushed, the angular position increases at a constant rate for the majority of the push, except for a short period of slower increase at the beginning and end of the push, as indicated in Figure 3. In order to create a basis for predicting the motion when a push occurs, the position, velocity, and acceleration of the first rise in the training data (from time zero to the first peak) is saved and will be referred to as the "push data". The push data is shown in Figure 5. The pendulum link is only pushed counterclockwise when feedback is given, which the encoder reads as a negative direction, therefore the push data is saved such that the values are negative.

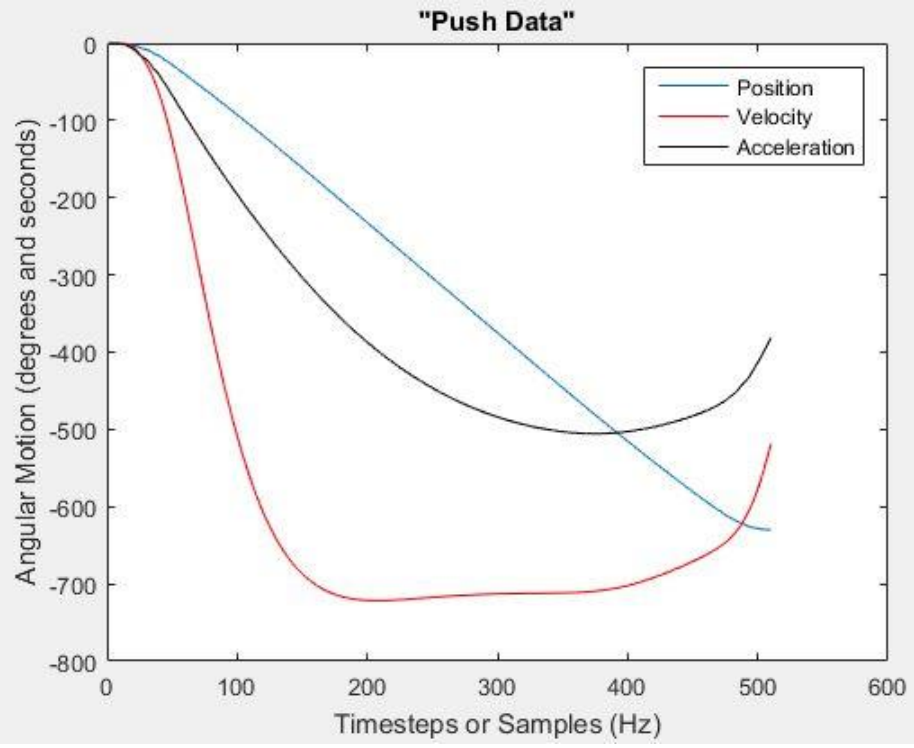

Figure 5: *The push data shown here is assumed to describe the motion of the pendulum whenever it rotates. That is, when the pendulum rotates, its position profile mainly changes at a constant rate, with a slower change at the start and end of the period of motion. The resulting velocity and acceleration from this position profile are also shown.*

For every time frame in which a push occurs (based on the voltage drop across the force sensor), a new "push" instance is generated in the MATLAB code, with a length corresponding to the number of timesteps, or samples (in Hz), for which the voltage consecutively remained within the required range. For each of these time frames, the push data shown in Figure 5 is scaled by a factor of (length of push)/(length of push data). The push data has 510 data points. Therefore, if a push is delivered for half as long (255 data points), the position, velocity, and acceleration for that push will be scaled to half the size of the push data, while retaining the same shape as the push data. An example of a scaling of the position to half the size of the push data position is shown in Figure 6.

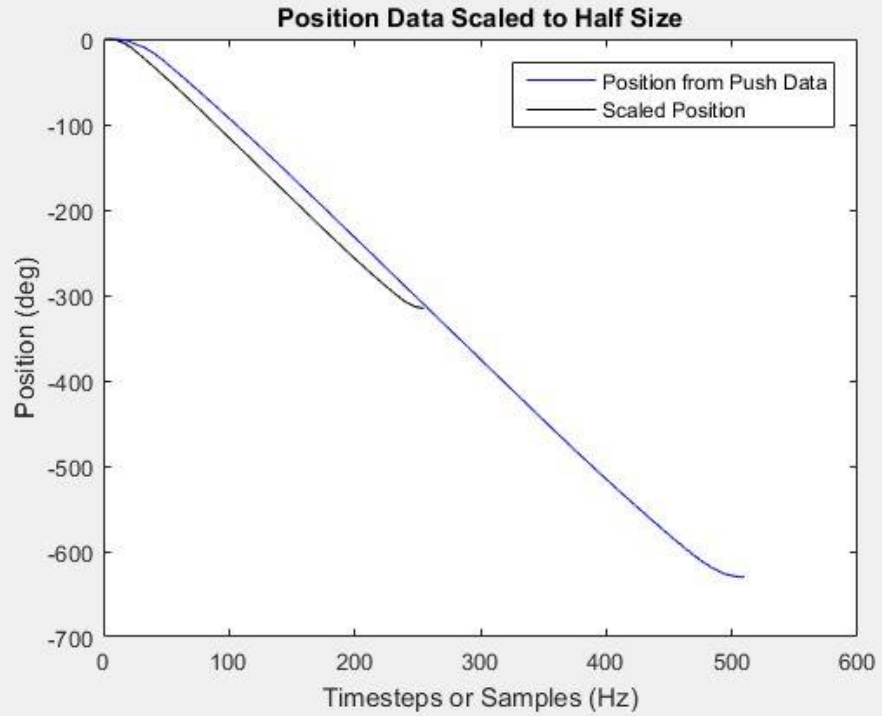

Figure 6: *If the length of time for which feedback is delivered is half of the length of the push data, the motion of the pendulum for that period is said to be the push data scaled by a factor of one-half, as shown. This scaling will allow the position profile to retain its three key features: a constant rate of change for most of the period, with a slower rate of change at the start and a slower rate of change at the end.*

Once the voltage is no longer in the required range, no push is occurring, therefore the values for position, velocity, and acceleration are all zero. While in reality the position value in such a case is equal to the position at the end of the previous push, there is no part of the training data during which the pendulum remains stationary. Therefore, the GPR model is unable to appropriately predict voltage required to keep the motor stationary (which should be zero volts). The only time during which the GPR model predicts a required voltage of zero is when position, velocity, and acceleration also have values of zero.

Figure 7 shows an example of what the predicted position, velocity, and acceleration from force feedback may look like. In Figure 8, the same position is shown with all stationary points of the motor having a position value of zero and positions for subsequent feedback periods starting from 0 (as explained in the preceding paragraph). This is the position data used for GPR querying. Since velocity and acceleration are already zero when the motor is stationary, these values do not have to be adjusted for GPR querying. The entire process for acquiring force feedback and using it to approximate the voltage required to produce the desired motion is shown in Figure 9.

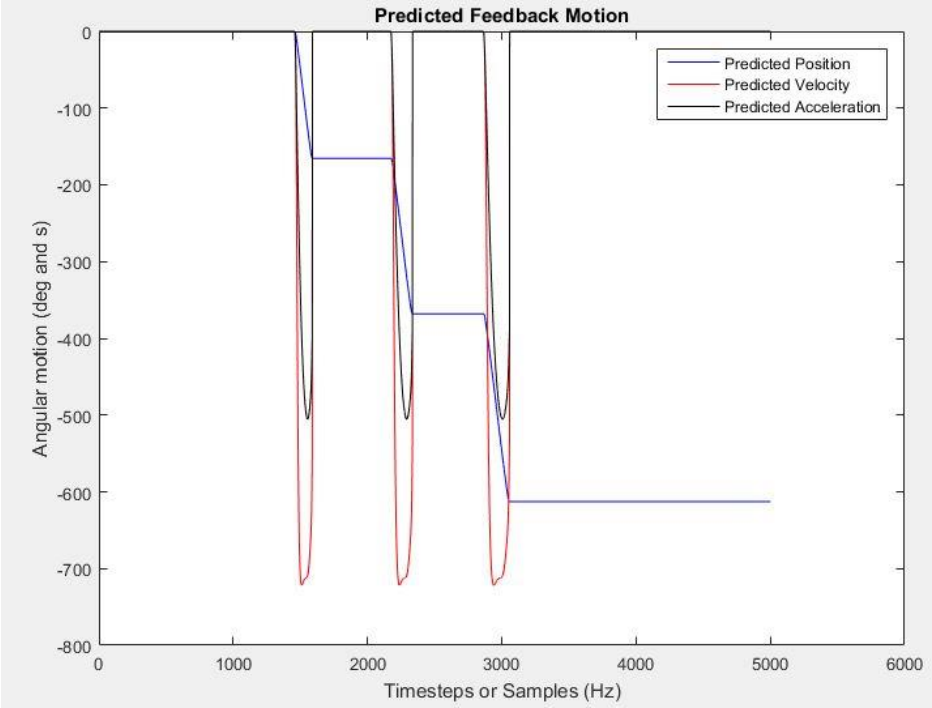

Figure 7: *When tactile feedback is given that causes the force sensor readings from Figure 4, the push data is scaled for each feedback period during which the sensor was pushed. The resulting motion that is predicted to have occurred due to the tactile feedback is shown here.*

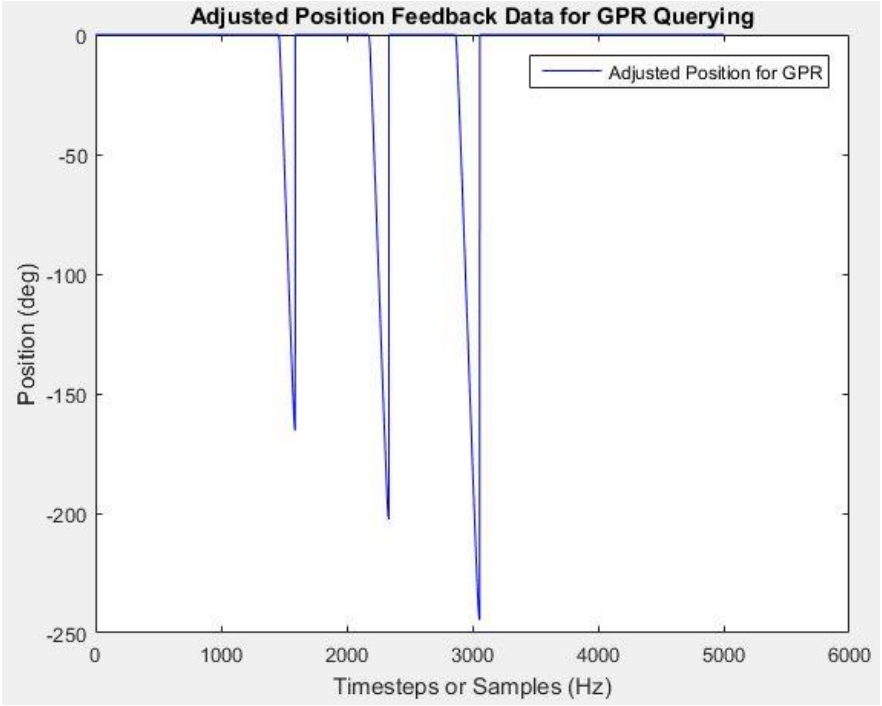

Figure 8: *Since the training data for the GPR model does not contain any periods of time during which the pendulum is stationary, the predicted position from the tactile feedback must be adjusted. Therefore, when the pendulum is stationary, the position is set as being zero and the next period of motion starts from zero rather than the last position.*

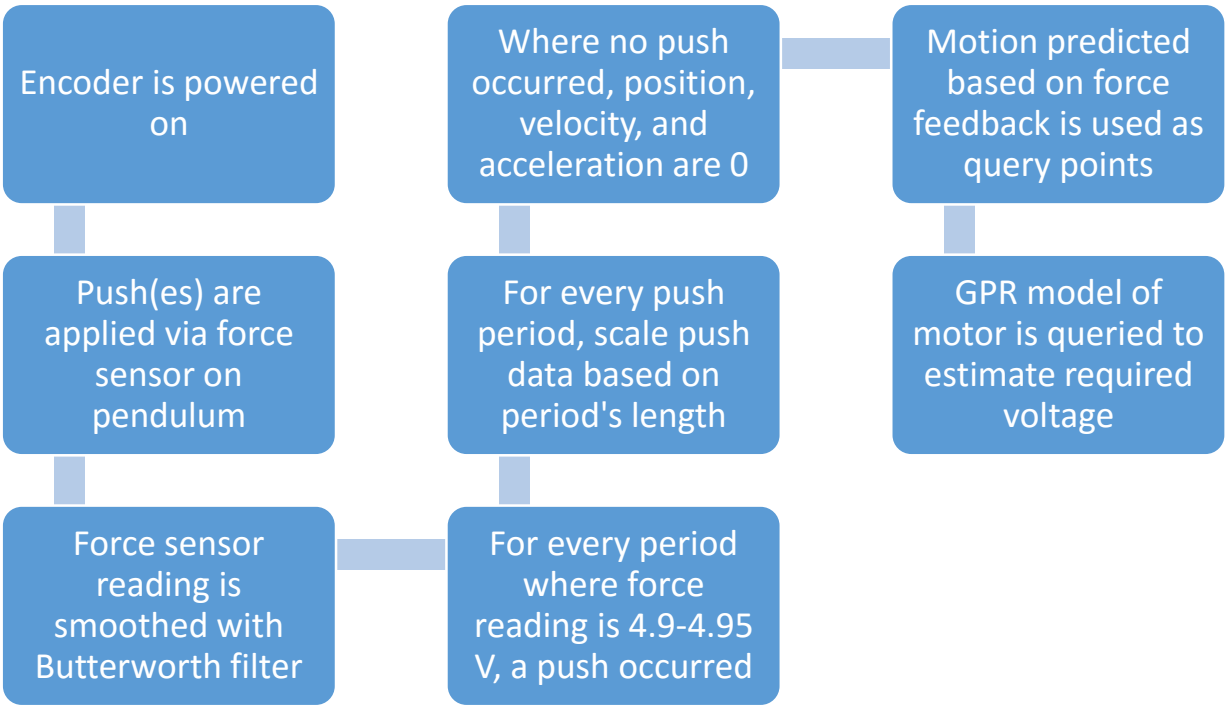

Figure 9: *Process for delivering tactile feedback to the pendulum and using that feedback to predict the desired motion and required voltages to achieve that motion. Notice the order of the steps in the flowchart, as indicated by the connections between the steps.*

4. Results

The results of the computer pendulum simulation are shown below. The learned model and the torques predicted for the query points are plotted in Figure 10, with the regression from the training data in the first half (0-3 seconds) and the predicted torques in the second half (3-6 seconds).

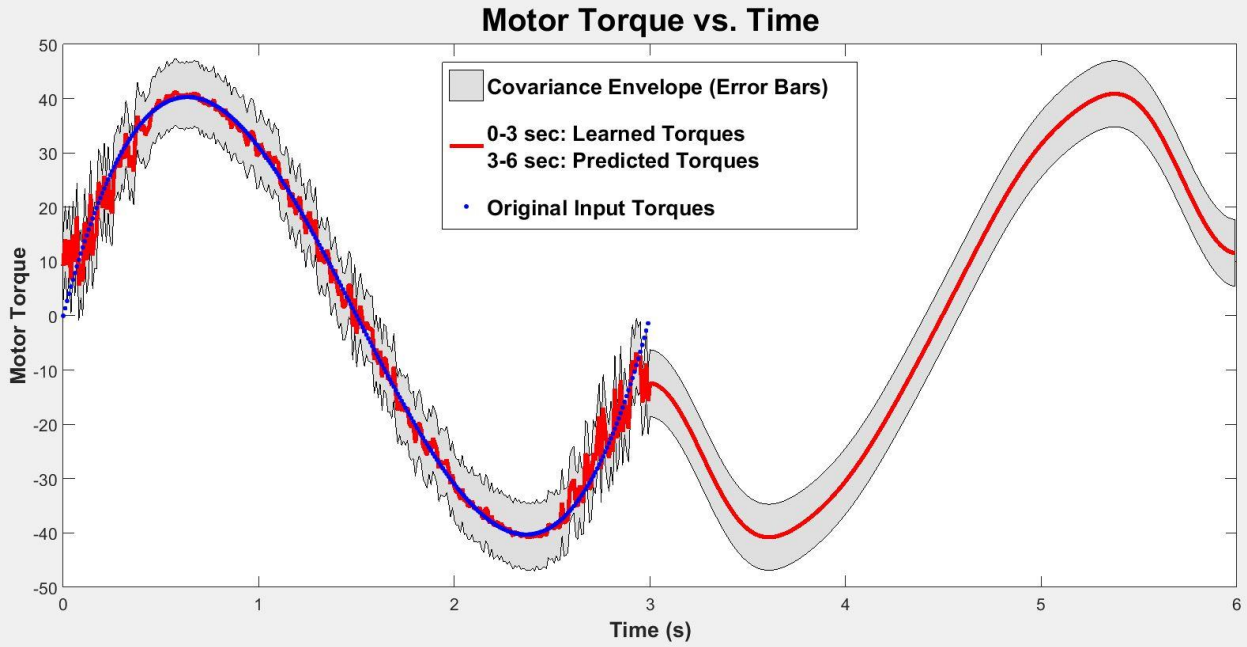

Figure 10: *The first half of this graph (0 to 3 seconds) shows the training torques while the second half (3 to 6 seconds) shows the torques predicted by the GPR model to reverse the motion of the pendulum. The gray area is the covariance envelope of the GPR predictions, shown in red.*

The resulting angular positions of the pendulum were then compared to the query points, as shown in Figure 11. The blue curve consists of the noisy training data for the first half (0-3 seconds) and the query (desired) points in the second half (3-6 seconds). The red curve consists of the resulting angular positions when the predicted torques were applied. As seen in Figure 11, the predicted torques resulted in angular positions very similar to the query points, indicating that GPR allowed proper tracking of the desired motion.

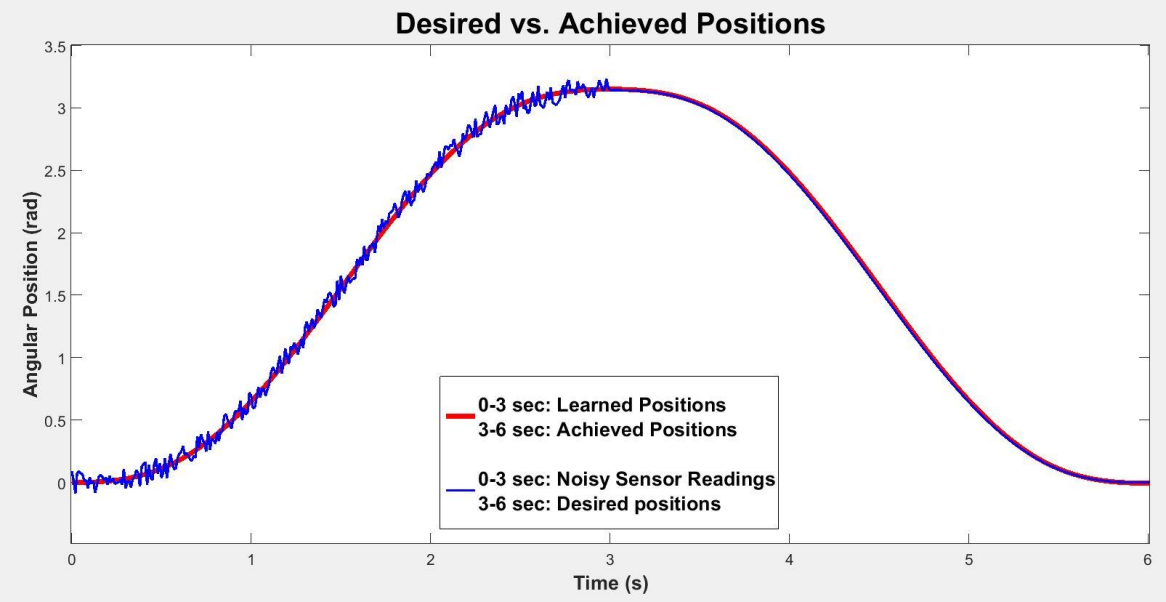

Figure 11: *The first half (0 to 3 seconds) shows the training positions (with noise) of the pendulum and the second half (3 to 6 seconds) compares the desired positions in blue to the achieved positions in red, due to the torque predicted by the GPR.*

Following are the results of the robotic pendulum proof of concept. Figure 12 shows the comparison between the desired position from when tactile feedback was given, in blue, and the predicted (or inferred) position based on the force sensor readings. Based on this predicted position profile, in red, the GPR model of voltage and motion was queried. In black is shown the position which was achieved when the resulting voltage predicted from the GPR model is applied to the motor. Figure 13 shows the comparison between the predicted velocity based on the force sensor readings and the achieved velocity. Such a comparison could not be derived for the acceleration of the motor because the acceleration output was too noisy to adequately filter with a Butterworth filter. However, based on the results of the position and velocity outputs of the motor, it may be assumed that the achieved acceleration has a similar degree of accuracy compared to the predicted acceleration.

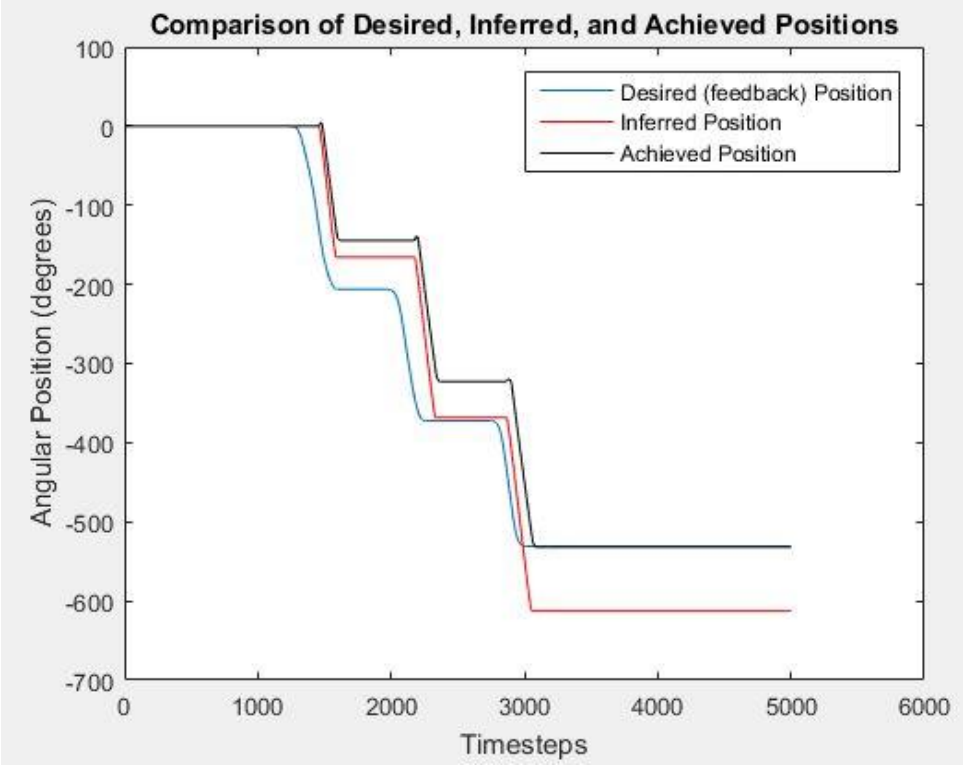

Figure 12: *Comparison of the desired position, which was recorded with the encoder during tactile feedback, the predicted or inferred position, based on the force sensor readings, and the achieved position resulting from the GPR predicted voltage.*

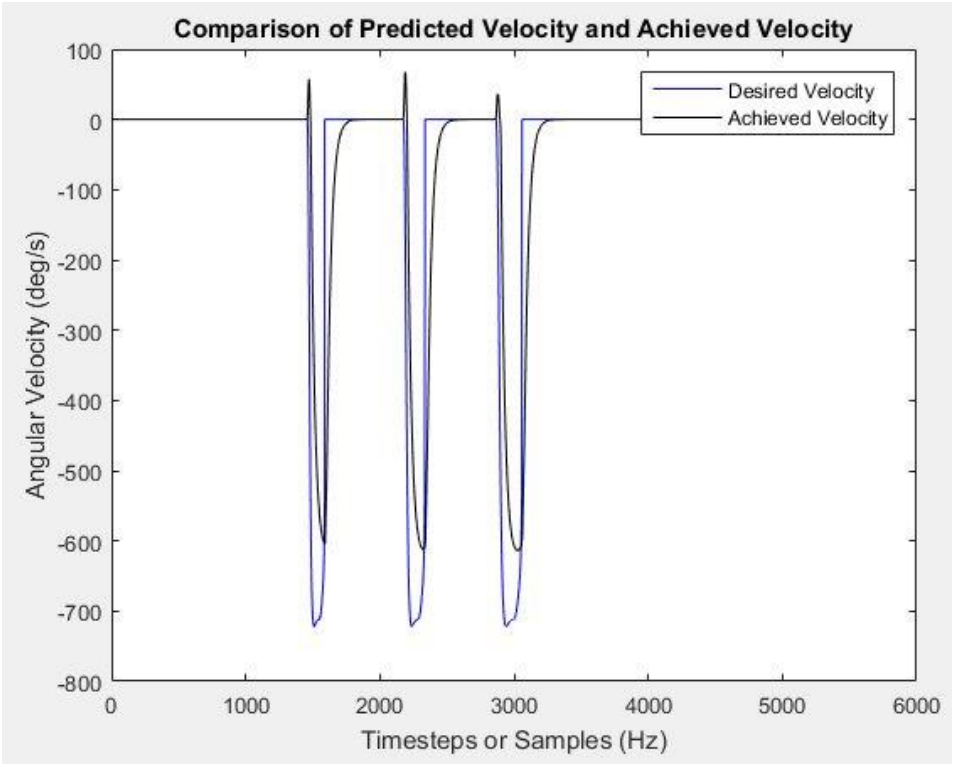

Figure 13: *Comparison of the desired velocity, predicted based on the force sensor readings, and the achieved velocity resulting from the GPR predicted voltage.*

An RMS error and percent error analysis of the position results is shown in Table 1. The fact that the final achieved position is identical to the final desired position seems to be coincidental, resulting from an over-prediction in the position (red) and an under-prediction in voltage (causing the achieved position to not resemble the predicted position exactly). Due to the low percent error between the desired and predicted position, the method for predicting the motion based on force feedback seems to be relatively accurate. The largest error is due to the GPR regression used to predict the required voltage based on the queried motion (predicted positions, velocities, and accelerations).

Table 1

Positions to Compare	*Desired vs Achieved*	*Desired vs Predicted*	*Predicted vs Achieved*
Average Percent Error	18.2%	3.7%	14.9%
RMS Error	55.6	64.0	57.0

In order to further test the feasibility of GPR for predicting the required voltage to produce a desired motion, new test motions were chosen as query points, without using force feedback. Two distinct examples of this are shown in Figures 14 and 15, where the queried positions are compared to the achieved positions. Figure 16 is the velocity comparison corresponding to the data in Figure 15.

Based on a visual comparison of the desired and achieved positions in Figure 14, it is apparent that the error is much less than when force feedback was used, as seen in Figure 12. In Figure 15, the error is even less, due to the fact that the queried motion was a subset of the training data. This is expected as the more the queried motion deviates from the training data, the less accurate the GPR prediction will likely be. Figure 15 is the most accurate (subset of training data with only one "step"), followed by Figure 14 (which takes two "steps"), followed by Figure 12 (three "steps").

Based on the results from both the tactile feedback and the follow-up investigation not using tactile feedback, it appears that GPR can be used to track a desired motion relatively well, though not without errors. The largest errors occur when the query points are the most different from the training data. Therefore, the more diverse and large a training data set is, the better the GPR regression will perform. Unfortunately, a large training set also becomes exponentially more computationally expensive since the covariance matrices are square matrices of size (length of training data) by (length of training data).

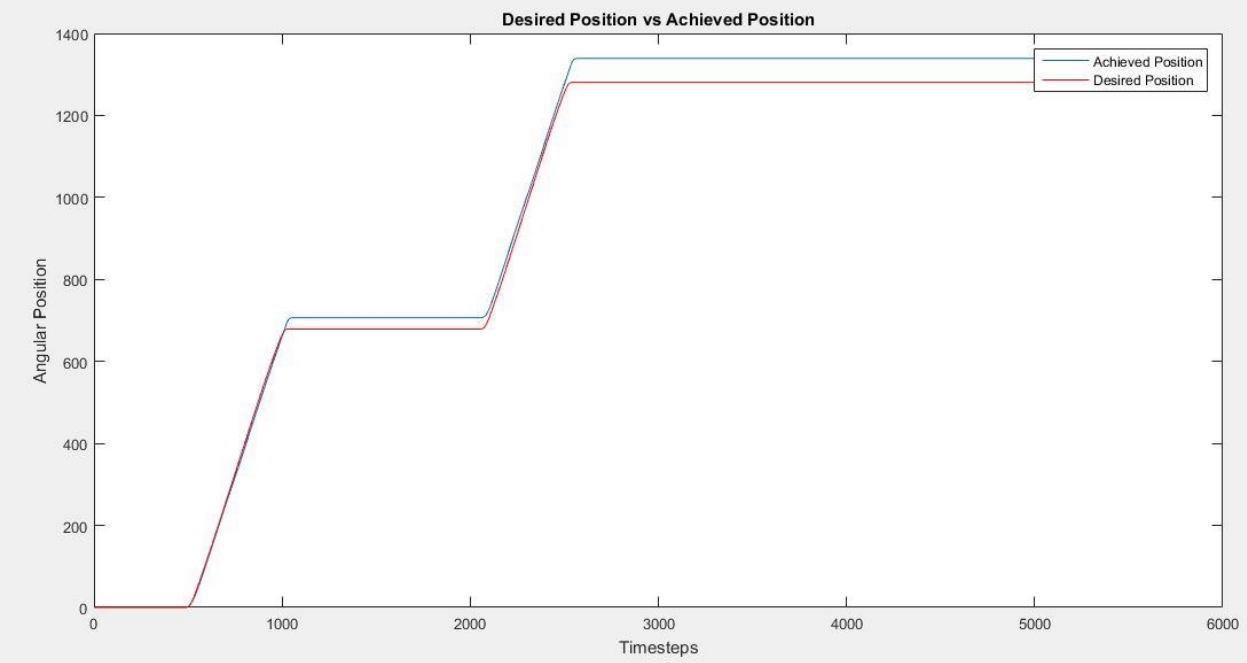

Figure 14: *Desired position used as query to GPR model versus achieved position caused by GPR predicted voltage. This is the two "step" example, where the desired positions were not acquired using tactile feedback.*

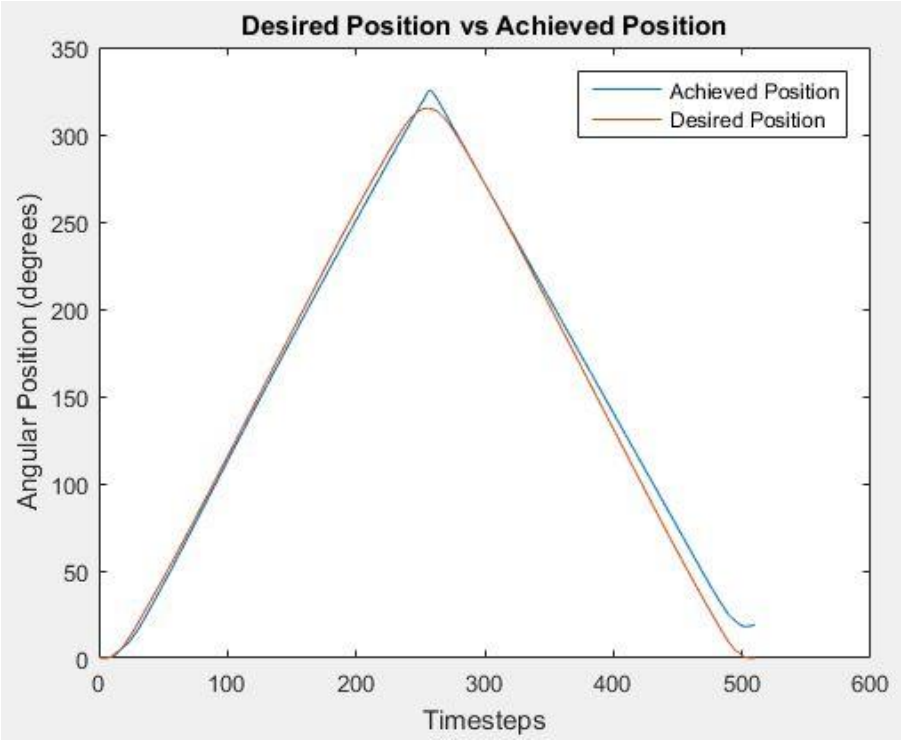

Figure 15: *Desired position used as query to GPR model versus achieved position caused by GPR predicted voltage. These desired positions are a subset of the training data and were not acquired using tactile feedback.*

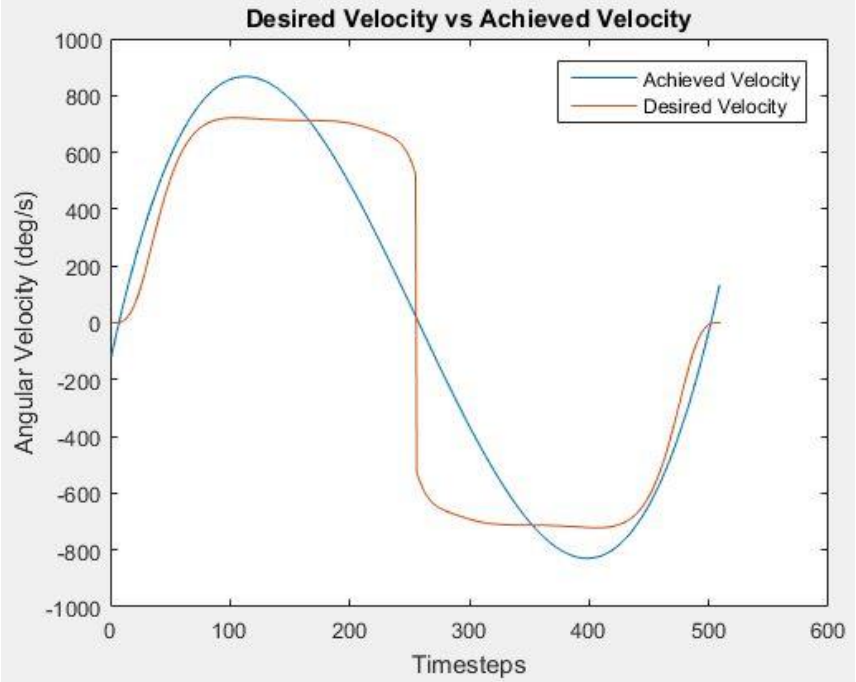

Figure 16: *Desired velocity used as query to GPR model versus achieved velocity caused by GPR predicted voltage. These velocities are produced from the same motion as the position data from Figure 15.*

Appendix

Gaussian Process Regression

The following discussion on machine learning and Gaussian Process Regression is written based on the work of Rasmussen and Williams (2006).

Machine learning is a broad field with many applications and methods of implementation. For the purposes of this project, supervised learning is used. Supervised learning has two steps: training an algorithm based on a set of labeled data, consisting of input/output pairs, and using that learned model to predict the resulting outputs of new inputs. The data is "labeled" because each output data point has an input data point associated with it. Unsupervised learning, by contrast, simply has an algorithm categorize data by differences and similarities, without a concrete understanding of what each data point means. Supervised learning is used in cases such as the one presented in this paper, where a specific outcome is desired (i.e.: moving a pendulum is a certain way). Unsupervised learning is generally used for data analytics (i.e.: creating a heat map of population density in a city).

The first step in supervised learning, training the algorithm, requires a set of training data to be recorded. This data is processed to find certain patterns between inputs and resulting outputs, thus creating a model of the system from which the data is gathered. Once this model is created, new input points (called query points) can be specified for which the algorithm predicts the corresponding outputs based on the previously learned system model.

While many machine learning algorithms exist, Gaussian Process Regression (GPR) was chosen for the problems presented in this paper. GPR is a form of Bayesian modeling which assumes a normal distribution of the data and defines the system by assigning means and covariances to the data. GPR is a preferred machine learning algorithm for regression problems such as the one presented in this paper and is therefore often used for situations involving motion control.

In a Gaussian process, it is assumed that the outputs of a system are related to its inputs by some function such that

$$f(x) = x^T w,$$

where x is a matrix comprised of the input vectors of the system and w is a vector of weights. In real-world systems, this relationship between the inputs and outputs of a system also includes some noise, which is assumed to have a Gaussian distribution with zero mean and a variance of σ_n^2

$$y = f(x) + \mathcal{N}(0, \sigma_n^2).$$

A set of inputs, x, and corresponding outputs, y, makes up the training data used to perform the regression. The joint distribution of the training points, x, and query points, x^*, can be written as

$$\begin{bmatrix} y \\ f^* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(x,x) + \sigma_n^2 I & K(x,x^*) \\ K(x^*,x) & K(x^*,x^*) \end{bmatrix}\right).$$

Here, y are the output points from the training data, f^* are the output points to predict given the query input points, I is the identity matrix, and K is a covariance function of the form

$$K\left(x^p,\ x^q\right)\ = \exp\left[-\frac{1}{2}\sum_{d=1}^{D}\frac{\left(x_d^p - x_d^q\right)^2}{w_d^2}\right],$$

where w_d represents hyperparameters such as length scale, signal variance, and noise variance. This leads to the main equations for performing predictions with Gaussian Process Regression, resulting in the predicted outputs, f^*, and the corresponding covariance envelope of the predicted outputs

$$f^* = K(x^*, x)[K(x, x) + \sigma_n^2 I]^{-1} y$$

$$covariance_{f^*} = K(x^*, x^*) - K(x^*, x)[K(x, x) + \sigma_n^2 I]^{-1} K(x, x^*).$$

References Cited

B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz. Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction. ACM, 2012.

C. H. Ho, R. J. Triolo, A. L. Elias, K. L. Kilgore, A. F. DiMarco, K. Bogie, A. H. Vette, M. Audu, R. Kobetic, S. R. Chang, K. M. Chan, S. Dukelow, D. J. Bourbeau, S. W. Brose, K. J. Gustafsin, Z. Kiss, and V. K. Mushahwar. Functional electrical stimulation and spinal cord injury. Physical Medicine and Rehabilitation Clinics of North America, 25(3):631–654, 2014.

W. D. Memberg, K. H. Polasek, R. L. Hart, A. M. Bryden, K. L. Kilgore, G. A. Nemunaitis, H. A. Hoyen, M. W. Keith, and R. F. Kirsch. Implanted neuroprosthesis for restoring arm and hand function in people with high level tetraplegia. Archives of Physical Medicine and Rehabilitation, 95(6):1201–1211, 2014.

Rasmussen, Carl Edward., and Christopher K. I. Williams. Gaussian Processes for Machine Learning. Cambridge, MA: MIT, 2006.

E. M. Schearer, Y. Liao, E. J. Perreault, M. C. Tresch, W. D. Memberg, R. F. Kirsch, and K. M. Lynch. Multi-muscle FES force control of the human arm for arbitrary goals. IEEE Transactions on Neural Systems and Rehabilitation Engineering, 22(3):654–663, 2014.

Sciavicco, Lorenzo, and Bruno Siciliano. Modelling and control of robot manipulators. Springer Science & Business Media, 2012.

B. Smith, P. H. Peckham, M. W. Keith, and D. D. Roscoe. An externally powered, multichannel, implantable stimulator for versatile control of paralyzed muscle. IEEE Transactions on Biomedical Engineering, 34(7):499–508, 1987.