Online Trajectory Generation Strategies for Needle-Based Interventions

Chris Morley,

Supervisor: Patel, Rajni V., The University of Western Ontario
A thesis submitted in partial fulfillment of the requirements for the Master of Engineering Science degree in Electrical and Computer Engineering
© Chris Morley 2023

Follow this and additional works at: https://ir.lib.uwo.ca/etd

Part of the Controls and Control Theory Commons

Recommended Citation
https://ir.lib.uwo.ca/etd/9549

This Dissertation/Thesis is brought to you for free and open access by Scholarship@Western. It has been accepted for inclusion in Electronic Thesis and Dissertation Repository by an authorized administrator of Scholarship@Western. For more information, please contact wlswadmin@uwo.ca.
Abstract

Needle-based interventions such as brachytherapy are among the most common minimally invasive procedures performed. Despite the numerous advantages of such procedures, surgeons are met with an array of challenges, most significantly, determining a strategy in real time to compensate for needle deflection as the needle passes through various layers of tissue, all having different mechanical properties. This thesis focuses on exploring new state estimation and control strategies to enhance the quality of needle-based interventions. These strategies include the use of machine learning for path planning and state estimation, while congruently exploring how the shape of a deflected needle can be used to explore reachable needle trajectories. Results and limitations are presented for the proposed strategies. A particular focus is made so that the strategies find a needle manipulation strategy that requires as few manipulations as possible.

Keywords

Summary for Lay Audience

The work presented in this thesis is focused on exploring ways in which the quality of care for patients undergoing a needle-based intervention can be improved. Common examples of these procedures include brachytherapy and muscle biopsies. Despite all the training a surgeon must do, needle-based interventions are still difficult for surgeons to perform. Human tissue is highly elastic, making it incredibly difficult to predict how a needle will pass through the tissue. Needles will deflect while travelling through tissue, sometimes resulting in the needle missing its target. In such an event, the surgeon may need to retract the needle and perform the insertion again until he/she is able to reach the target location. Excessive needle insertions may cause patient discomfort and tissue damage leading to longer recovery times. This thesis explores methods of predicting how a needle will travel through tissue. The work presented is intended to be carried out using robotics to perform needle manipulations.

Much of the work done for needle-based interventions focuses on finding analytic models to describe the needle-tissue interaction during a procedure. Parameters in these models include friction, cutting force, clamping force, expected curvature and many more. An issue arises when using analytical models because there are always going to be unmodelled dynamics that are not captured by the proposed model. This thesis focuses on adaptive strategies that use the shape of the needle and modern machine learning techniques to predict the trajectory of a needle in response to manipulations of the needle performed using a robotic arm.
Co-Authorship Statement

The entirety of this thesis was written by Chris Morley as a requirement for the Masters of Engineering Science Degree at Western University. The work was carried out at Canadian Surgical Technologies and Advanced Robotics (CSTAR) lab. The research work was carried out under the supervision of Dr. Patel, who offered guidance and support throughout the duration of the research. Fellow lab member Anirudh Vajpeyi, and lab alumni Anish Naidu helped assemble electromagnetic trackers and securely embed them within a needle for experimental work. Anish Naidu also helped develop a C++ interface for the force/torque sensor used in the experimental setup.

Material presented in sections 4.2 through 4.4, has been published in the 2022 RAAI Conference Proceedings:

Acknowledgments

First and foremost, I must express my most sincere gratitude to Dr. Patel for welcoming me into his lab and offering his guidance and supervision over the past few years. Dr. Patel always keeps his students’ best interests in mind, ensuring that while under his supervision they gain the skills needed to pursue their goals. He not only provided me with all the resources needed to pursue my research but also suggested alternative methods of pursuing my research in such a way that I could study new emerging topics in research. As my research came to an end, Dr. Patel was incredibly supportive when I approached him about entering the workforce through an exciting internship opportunity available; he not only encouraged me to pursue it but offered advice on how to best present my newfound skills from grad school. Dr. Patel has been instrumental in furthering my academic career and aiding in the start of my professional career.

My time spent at Western was without a doubt one of the most fun periods of my life. Looking back, it is hard to imagine there was a time in my life when I had not yet met the friends I have made at Western. I look forward to making new memories with all of you.

Most importantly I would like to thank my parents for the love and support they have given me over the years. The support I have gotten from my parents has enabled me to pursue a career in a field I love. To be able to go to work every day and work on something I am truly passionate about is the greatest gift of all. I hope one day to be able to provide the same opportunities to my kids. I can say with confidence that I certainly would not be where I am today without the support of my parents; so to you both, I thank you.
# Table of Contents

Abstract .................................................................................................................. ii

Summary for Lay Audience .................................................................................... iii

Co-Authorship Statement ......................................................................................... iv

Acknowledgments ...................................................................................................... v

Table of Contents ..................................................................................................... vi

List of Tables .......................................................................................................... ix

List of Figures .......................................................................................................... x

List of Acronyms ..................................................................................................... xv

Chapter 1 .................................................................................................................. 1

1 Introduction .......................................................................................................... 1

1.1 Motivation .......................................................................................................... 1

1.2 Prostate Cancer Treatment .............................................................................. 1

1.2.1 Surgery .......................................................................................................... 2

1.2.2 Chemotherapy ............................................................................................ 2

1.2.3 Brachytherapy ............................................................................................ 3

1.3 Literature Review ............................................................................................. 5

1.3.1 Kinematic Modelling .................................................................................. 6

1.3.2 Needle-Tissue Interaction Force Modelling ................................................. 7

1.3.3 Machine Learning in Needle-based Interventions ..................................... 8

1.3.4 Current Gap in Robotic Needle-based Interventions .................................. 9

1.4 Contributions .................................................................................................. 10

Chapter 2 .............................................................................................................. 11

2 State Estimation and Controller Design ............................................................ 11

2.1 State Estimation Review ............................................................................... 12


<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5.2</td>
<td>Savitzky-Golay Closed-Loop Performance</td>
<td>73</td>
</tr>
<tr>
<td>Chapter 4</td>
<td></td>
<td>77</td>
</tr>
<tr>
<td>4</td>
<td>Steering of Flexible Needles using an LSTM Encoder with Model Predictive Control</td>
<td>77</td>
</tr>
<tr>
<td>4.1</td>
<td>Machine Learning Review</td>
<td>77</td>
</tr>
<tr>
<td>4.2</td>
<td>Introduction</td>
<td>79</td>
</tr>
<tr>
<td>4.3</td>
<td>Methodology</td>
<td>82</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Collecting Experimental Data for Modeling</td>
<td>82</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Needle Insertion Model</td>
<td>82</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Dataset Generation</td>
<td>84</td>
</tr>
<tr>
<td>4.3.4</td>
<td>Trajectory Prediction Network</td>
<td>85</td>
</tr>
<tr>
<td>4.3.5</td>
<td>Trajectory Prediction with Rotation Network</td>
<td>88</td>
</tr>
<tr>
<td>4.3.6</td>
<td>Extrapolating Prediction Horizon</td>
<td>90</td>
</tr>
<tr>
<td>4.3.7</td>
<td>Model Predictive Controller</td>
<td>92</td>
</tr>
<tr>
<td>4.4</td>
<td>Simulation Results</td>
<td>93</td>
</tr>
<tr>
<td>4.5</td>
<td>Experimental Results</td>
<td>96</td>
</tr>
<tr>
<td>Chapter 5</td>
<td></td>
<td>104</td>
</tr>
<tr>
<td>5</td>
<td>Conclusion and Future Work</td>
<td>104</td>
</tr>
<tr>
<td>5.1</td>
<td>Summary</td>
<td>104</td>
</tr>
<tr>
<td>5.2</td>
<td>Future Work</td>
<td>106</td>
</tr>
<tr>
<td>References</td>
<td></td>
<td>108</td>
</tr>
<tr>
<td>Chris Morley Curriculum Vitae</td>
<td>115</td>
<td></td>
</tr>
</tbody>
</table>
List of Tables

Table 4.1: Deflection Model Curve Fitting................................................................. 83
List of Figures

Figure 1.1 – Diagram of a LDR procedure ................................................................. 4

Figure 2.1 - Kalman Filter Algorithm ................................................................. 12

Figure 2.2 - Control Polygon Interpolation ......................................................... 19

Figure 2.3: Piecewise function with non-continuous second derivative ................. 25

Figure 2.4: Illustration of shorter filter length producing better derivative estimates when the signal of interest does not follow steady state dynamics ........................................... 26

Figure 2.5 – Adaptive filter length LSTM Network ............................................. 28

Figure 2.6 – Optimal filter length prediction with LSTM network ..................... 29

Figure 2.7: Convergence of LSTM predicted RMSE and SD of RMSE .................. 29

Figure 2.8: Convergence of the ratios comparing adaptive and optimal filter lengths with fixed filter length .............................................................. 30

Figure 2.9: Convergence of the ratios comparing first and second derivative estimates of LSTM SG-Filter with Kalman Filter .............................................................. 32

Figure 2.10: Comparison between adaptive LSTM SG filter and Kalman filter second derivative estimate .............................................................. 33

Figure 2.11: Needle base and tip frame ................................................................. 35

Figure 2.12: Trajectory prediction block diagram using an LSTM SG filter .......... 38

Figure 2.13: Trajectory prediction method #2 - block diagram ......................... 39

Figure 3.1: Kinova Gen2 6 degrees-of-freedom Robotic Manipulator .................. 42

Figure 3.2: Exploded view of CAD model of the custom-designed end-effector and needle mount ..................................................................................... 45
Figure 3.3: 3D printed end-effector that houses the ATI force sensor for measuring reaction forces at the base of the needle ................................................................. 46

Figure 3.4: Aurora NDI tabletop field generator [43]................................................................. 47

Figure 3.5: Aurora NDI System Control Unit (SCU) ................................................................. 47

Figure 3.6: EM tracker shown outside of the hollow needle prior to fixing a sensor within the needle ............................................................................................................. 51

Figure 3.7: EM trackers inside the hollow needle ........................................................................ 51

Figure 3.8: Experimental Testbed Setup ..................................................................................... 52

Figure 3.9: Experimental Setup Architecture .............................................................................. 53

Figure 3.10: Experimental Setup Control Graphical User Interface (GUI) ............................... 53

Figure 3.11: Simulink controller state machine ............................................................................ 54

Figure 3.12: Simulink needle controller inputs and outputs ...................................................... 55

Figure 3.13: Kinova Gen2 logic block diagram. This block diagram includes workspace boundary checking, target trajectory switching and cartesian to joint-space velocity mapping. ............................................................................................................. 56

Figure 3.14: Simulink custom library blocks for hardware ...................................................... 57

Figure 3.15: Simulink block diagram for autonomous data collection (Part 1) ................. 58

Figure 3.16: Simulink block diagram for autonomous data collection (Part 2) ...................... 59

Figure 3.17: Sample of needle tip velocity estimate using Bezier Curve tangent (X-Cartesian component of velocity) ............................................................................................................. 62

Figure 3.18: Comparison between Kalman filter, with and without BC tangent as velocity estimate. SD shown with error bars. ............................................................................................................. 62

Figure 3.19: Poor quality trajectory prediction ........................................................................... 63
Figure 3.20: Needle-Tip Error vs Target and Prediction Depth ........................................ 64
Figure 3.21: Needle-Tip Position Error vs Prediction Depth and Rotation Angle .............. 65
Figure 3.22: Needle-Tip Position Error vs Target Depth and Rotation Angle .................. 65
Figure 3.23: Needle-Tip Position Error vs Prediction and Target Depth ....................... 66
Figure 3.24: Needle-Tip Position Error vs Rotation Angle and Prediction Depth ............ 67
Figure 3.25: Needle-Tip Position Error vs Rotation Angle and Target Depth ................. 67
Figure 3.26: Trajectory Prediction with no rotation using Bezier Curve method. ............ 68
Figure 3.27: Trajectory prediction with rotation using Bezier Curve method .................. 68
Figure 3.28: Needle-tip position error using the SG-Filter over a large range of target and prediction depths ................................................................. 69
Figure 3.29: Needle-tip prediction error surface using the SG-Filter on a restricted domain of prediction depths ................................................................. 70
Figure 3.30: Needle-tip prediction error using the SG-Filter on restricted prediction depth domain. The surface is plotted against needle rotation and prediction depth .............. 70
Figure 3.31: Needle-tip prediction error surface using the SG Filter on restricted prediction depth domain. The surface is plotted against needle rotation and target depth .......... 71
Figure 3.32: Histogram showing distribution of final position errors when the acceptable threshold is set to 3 mm. ................................................................. 72
Figure 3.33: Scatter plot of the final position errors plotted against the optimal angle determined by the optimizer. The data points of zero rotation are instances where no solution was found ................................................................. 73
Figure 3.34: Histogram showing the distribution of the final position errors when the acceptable threshold is set to 3 mm. ................................................................. 74
Figure 3.35: Scatter plot of the final position errors plotted against the optimal angle determined by the optimizer. ................................................................. 75

Figure 3.36: Velocity and acceleration estimates using the adaptive LSTM-SG filter. ....... 75

Figure 3.37: Demonstration of the misalignment transforming the deflected needle-tip measurements into the end-effector frame.................................................. 76

Figure 4.1: Vanilla RNN cell ............................................................................. 77

Figure 4.2: Unrolled RNN ................................................................................. 78

Figure 4.3: An LSTM cell................................................................................. 78

Figure 4.4: Frames for the Needle Kinematic Model ......................................... 83

Figure 4.5: The Trajectory Prediction Network................................................... 87

Figure 4.6: Latent Space Transformation Network ............................................. 89

Figure 4.7: Trajectory Prediction with Needle Manipulation Network............... 90

Figure 4.8: Trajectory Prediction Demonstration .............................................. 94

Figure 4.9: Closed Loop Performance with Trajectory Prediction (a) Bezier curve (blue) is used to extrapolate prediction horizon (b) Final trajectory of needle......................... 95

Figure 4.10: Example of position data from a sequence with no rotation used to train an LSTM network .............................................................................................. 98

Figure 4.11: Example of position data from a sequence with rotation used to train an LSTM network .............................................................................................. 98

Figure 4.12: Poor performance of the network predicting future needle trajectory in the scenario where no rotation was performed ......................................................... 99

Figure 4.13: Good performance of the network predicting future needle trajectory in the scenario where no rotation was performed ......................................................... 99
Figure 4.14: Example of very good network prediction quality using simulation data

Figure 4.15: Training progress of the network trained using experimental data from 400 insertions.

Figure 4.16: Training progress of the network trained using experimental data from 800 insertions.

Figure 4.17: Training progress of the network trained using experimental data from 1300 insertions.

Figure 4.18: Training progress of the network trained with simulation data from 1000 insertions.
List of Acronyms

LDR: Low Dose Radiation
FEA: Finite Element Analysis
RMSE: Root Mean Square Error
SD: Standard Deviation
RNN: Recurrent Neural Network
LSTM: Long-Short Term Memory
EE: End Effector
KF: Kalman Filter
SG: Savitzky-Golay
BC: Bezier Curve
EM: Electromagnetic
CAD: Computer-Aided Design
API: Application Programming Interface
GUI: Graphical User Interface
F/T: Force/Torque
Chapter 1

1 Introduction

1.1 Motivation

Needle-based percutaneous interventions are perhaps the most common medical intervention, performed in many domains including local anesthesia, blood samples [1], tissue biopsies [2], neurosurgery [3], and internal radiation therapy [4]. These interventions are widely preferred among medical professionals for a variety of reasons, one of the most notable being the minimally invasive nature of the procedure. Many positive outcomes of needle-based interventions can be attributed to their minimally invasive nature, including reduced tissue damage, faster recovery times, greater patient comfort and reduced risk of infection [5]. Two specific, and very common, percutaneous procedures include needle biopsies and brachytherapy, with the objective of the former being to extract muscle fibers from a particular location in the body and the later focused on depositing radioactive capsules to a particular location within the body. The commonality shared between these procedures is the requirement to accurately guide the needle to the intended target within the patient’s tissue. While these two procedures are often favoured by medical professionals, they are not easy to perform given the highly complex nature of the interaction between a needle and patient tissue. These interventions often require the needle to pass through several different layers of tissue including skin, fat muscle and even connective tissue, all varying in mechanical tissue properties [6]. The vital role of percutaneous procedures is best recognized by reviewing the alternative treatment options available to a patient diagnosed with prostate cancer, a common illness.

1.2 Prostate Cancer Treatment

Prostate cancer is the most common form of cancer among Canadian men, and the third leading cause of death in that same demographic. It is expected that 1 in 8 Canadian men will be diagnosed with prostate cancer during their lifetime and 1 in 29 will die from it [7]. Should it be determined that the state of one’s cancer is severe enough to warrant
intervention, there are a variety of treatment options available. Most treatment options can be classified as either surgery, chemotherapy, or brachytherapy.

1.2.1 Surgery

Surgery is the most invasive class of interventions. The hopeful outcome is for surgeons to remove the cancerous tissue from the body by means of cutting it away. Because the mechanism of cancer spreading requires only a single cancer cell to multiply, surgeons must take great care to ensure that all the cancerous tissue is removed to prevent recurrence. This includes carefully removing the cancerous tissue so that it does not contact any other parts of the body as it is removed. If a biopsy was performed as part of the diagnosis, the tissue where the biopsy needle passed must also be removed to prevent the recurrence of cancer. Removing non-cancerous tissue that is in direct contact with cancerous tissue is referred to as leaving a surgical margin. The main upside with open surgery is that often the surgeon(s) can directly manipulate the cancerous tissue with a non-obstructed view of the tumor. Open surgery, however, may not be a viable option for several reasons including the likelihood of damaging surrounding organs or the health status of the patient. As with any open surgery, there is an increased risk for infection. Further, the recovery times and patient discomfort are greatly increased when compared to other less invasive procedures.

1.2.2 Chemotherapy

Chemotherapy is the delivery of drugs to parts of the body with the intent to kill cancerous cells. The drugs used target all cells but are particularly effective in killing cells that multiply at a fast rate, such as cancerous cells. These drugs work by further damaging the DNA of cancerous cells so that they can no longer reproduce. Examples of these drugs used include DNA-damaging agents, Antimetabolites and Antitumor antibiotics. The main drawback with chemotherapy is that these drugs cannot selectively target cancerous cells. Other cells which multiple at a rate similar to cancer cells, and thus are targeted by these drugs, includes cells such as those which make up hair follicles and the lining of the digestive tract. This is the reason that patients undergoing chemotherapy often experience hair loss and suffer from nausea and loss of appetite.
1.2.3 Brachytherapy

Brachytherapy is a type of internal radiation therapy (IRT) where hollow needles are used to place radiation emitting vessels, called pellets, inside a patient to deliver radiation locally to a tumor. The strength of the radiation attenuates rapidly with respect to the distance from the pellet [8]. This type of procedure is advantageous because it address the limitations of surgery and chemotherapy by being able to selectively target cells in a minimally invasive manor. There are two types of brachytherapy, high-dose rate (HDR) and low-dose rate (LDR) [9]. HDR places the pellets inside a patient for a duration on the order of a few minutes before the pellet is retracted through the needle, whereas LDR deposits the pellets within the patient permanently [10]. This thesis focuses on LDR brachytherapy procedures.

Traditionally, brachytherapy has been performed manually by surgeons. When carried out manually, a surgeon maneuvers a needle such that the tip is steered towards the center of a tumor. This needle is hollow, allowing the surgeon to deposit radioactive pellets (iodine-125 or palladium-103) through the needle into the tumor [11]. Pellets are approximately the same size as a grain of rice. For the brachytherapy procedure, the patient is positioned in a supine position on an operating table with their feet up on supports hanging off the end of the table [12]. A guide with holes in a grid arrangement is mounted to the end of the table, nearest the patient’s rectum. For a brachytherapy treatment, the amount of radiation and the placement of the radioactive pellets are first determined preoperatively. Based on this, the best way to reach the target for each pellet (i.e., the choice of the holes) is planned. The surgeon then begins the procedure by inserting needles through the grid holes under ultrasound (US) guidance (using a US rectal probe) [13]. For each needle insertion, the surgeon applies force to the needle to drive the needle into the prostate. The grid is used to help the surgeon with both placement and orientation of the needle while also preventing excessive lateral force on the tissue at the site of insertion. Ultrasound (US) imaging is used to observe the location of the needle tip as it is inserted into the prostate. The surgeon manipulates the needle as required to steer the needle to the desired location. The location may be within the volume of a tumor or may be within a close proximity to the tumor, depending on the radiation treatment strategy. The surgeon may repeat this process as many times as needed to insert the required number of pellets. Figure 1.1 illustrates the procedure.
The success of this procedure is directly correlated to the proximity of the pellet’s final location with respect to the target location. Difficulties in accurately placing the pellets arise from both aligning the needle, prior to puncturing the skin, and steering the needle while within tissue. Human tissue has very nonlinear elastic properties which make the trajectory of a needle very difficult to predict and control. A typical path that a needle will follow to get to a tumor located in the prostate will involve the needle passing through various layers of different tissue including skin, fat, and muscle, all of which having different mechanical properties. Bevel tip needles are commonly used for brachytherapy as their asymmetric shape provides a more controllable trajectory while slicing through tissue. Surgeons often employ a strategy of rotating the needle by 180 degrees at various points throughout the insertion to compensate for the accumulated deflection that can be attributed to the bevel tip shape. This strategy has drawbacks as rotation causes a drilling effect between the needle and tissue which damages tissue [14]. Despite the challenges present in performing a successful percutaneous needle-based intervention, the outcome of a successful procedure has enormous benefits when compared to traditional methods such as surgery or chemotherapy, this has driven the development of needle-tissue modeling and control strategies since the early 2000s.

Figure 1.1 – Diagram of a LDR procedure
1.3 Literature Review

As discussed above, there are many challenges associated with percutaneous procedures. One subset of challenges, although not the focus of this thesis, surround the surgeon’s obstructed view of the needle and tumor during the procedure [15]. Various imaging techniques used include X-ray, computed tomography (CT), magnetic resonance imaging (MRI), and ultrasound (US). These imaging techniques present challenges. X-rays and CT scans emit levels of radiation far too high to be used for the length of a typical needle-based intervention, an MRI requires that all instruments be non-ferromagnetic [16], and ultrasound imaging requires that the probe orientation and position be continuously monitored to track the needle. Another very challenging aspect of subcutaneous procedures, and the difficulty area of focus for this thesis, is the ability to make accurate predictions of a needle trajectory in response to manipulations of the needle at its current location, all while in the presence of varying tissue properties. Each of the layers (skin, fat, muscle, connective tissue) have different elastic properties that vary person-to-person for a variety of reasons including age, health-conditions, prior injuries, and body composition [17]. Further, the elastic properties of all tissue types mentioned are highly non-linear and anisotropic [18]. These tissue properties present a challenge to surgeons trying to accurately steer needles along a preplanned trajectory [19], let alone predict the future trajectory. To make matters more challenging, subcutaneous needle insertion is further complicated in many regions of the body where physiological motions generated by organs exist, adding exogenous relative motion between the needle and surrounding tissue [20]. Given the many challenges that must be overcome to successfully complete a subcutaneous needle insertion, lots of effort has been put towards developing needle-tissue models and control strategies for performing needle-based interventions. These advanced control techniques are not feasible for a human to perform, so researchers have turned to using robotics to carry out high precision needle manipulations. Robotic systems have certain attributes that are advantageous when compared to humans with regards to performing surgery. These attributes include improved precision, accuracy, attenuation of physiological movements, motion scaling, and increased degrees-of-freedom of tools within a patient [21] [22] [23]. Further, robotic systems can be setup directly with the imaging systems used to observe the progression of a procedure [24]. The remainder of
this chapter explores research to date surrounding needle-tissue interaction modeling, and control of subcutaneous needles manipulations using robotics.

1.3.1 Kinematic Modelling

One of the most impactful contributions early in the field of robotic needle insertion explored modeling the kinematics of a bevel tip needle traveling through a gelatin phantom tissue [25]. The work presented two models, namely the bicycle and unicycle models, both driven by linear and rotational velocity inputs. The bicycle model assumed the needle would follow a path analogous to a bicycle, having the angle of its steering wheel locked, while the unicycle model was a simplified version with a trajectory evolving with a constant radius. The authors performed data fitting to find a set of optimal model parameters from experimental data. Both models were shown to be capable of predicting the path of a needle with some degree of accuracy when inserted into a gelatin mold, however the bicycle model performed better by a statistically significant margin when compared to the unicycle model. The authors hypothesized these results as the bicycle model had more parameters and thus an increased capacity to fit a model accurately. This model, however, did have its shortfalls. The model required that the needle be of a comparable stiffness to that of the material which it was inserted into. This is a very restrictive requirement for any model which is to be used in a clinical setting as most procedures will require a needle to pass through different layers of tissue, such as skin, fat, and muscle, all having different elastic properties. Simulation results suggested that this model would fail to accurately model the needle trajectory in a gelatin mold of a different density. The shortcomings of this work can largely be attributed to the fact the interaction forces between the needle and tissue were neglected. There is an exchange of mechanical work between the needle and tissue during an insertion. Neglecting these interaction forces and modeling the needle-tissue as a non-coupled system is an insufficient description of the interaction and thus incapable of accurately modeling said interaction. The coupled behavior of the needle and tissue motivated other research groups to explore models that consider the needle-tissue interaction forces.
1.3.2 Needle-Tissue Interaction Force Modelling

DiMaio and Salcudean were among the pioneers of FEA models for soft tissue in the domain of needle insertion. Their work in [26] hypothesized that reaction forces at the proximal end of the needle are insufficient to estimate the shape of the needle as the shape depends on the integration of forces acting along its shaft. DiMaio and Salcudean performed experiments to track hundreds of points on the surface of a gelatin phantom while a needle was inserted into the phantom. The recorded data was used to develop a FEA model that could estimate the forces acting at the distal end and along the shaft of the needle. This FEA model was then used to develop a haptic simulation for training purposes. While these models have been shown to model needle deflection with reasonable accuracy, they do not generalize well to different patients and are too computation costly to run in a real-time control loop. Machine learning could greatly improve the usability of these FEA models by training networks to learn the deformation of the needle experiencing an internal load, while being far less computationally costly [27]. Mechanics-based models have been proposed to model needle deflection. Much of the work in mechanics-based models uses beam theory to model deflection along the shaft of the needle. Various groups have suggested models that use combinations of different types of common distributed loads, acting along the length of the needle. The Authors of [28] assumed the deflection of a needle was attributed to a point load acting at the needle tip and a triangular distributed load acting along the length of the needle. In [29], an adaptive quasi static method using uniform, triangular and quadratic distributed loads, in addition to a point load at the needle tip was proposed. An attractive feature of this work was that it mapped reaction forces at the base of the needle to estimated distributed loads which were then used to predict needle deflection. They used a numeric optimization to determine a mapping between reaction forces and coefficients of varying distributed loads, acting on needles from experimental data. One major drawback is that all these mechanics-based models require an assumption to be made about the shape and number of distributed loads when in reality, the loads acting on a needle do not adhere to these common distributions. The authors of [30] presented a mechanics-based model that was derived using the Rayleigh-Ritz method. This method uses the principle of minimum energy to determine the equilibrium state, i.e. shape of the needle, given the current load(s) acting on the needle. A novel contribution of this paper
was the ability to model a needle trajectory that had undergone multiple rotations. This is particularly beneficial in scenarios where extra flexible needles are used to execute highly curved trajectories for the purpose of steering around anatomical obstacles. A limitation of this work is that a linear approximation is made when computing the strain energy using Euler-Bernoulli beam theory, this approximation is only accurate for small angles. Additionally, the mechanical tissue properties are modelled as linear springs, lacking accurate representation of a clinical scenario. Another challenge when designing needle-tissue interaction models is that the model parameters are not always observable. The coupling between the needle and surrounding tissue imposes several different reaction forces on the needle including puncture, cutting, viscous/coulomb friction, and clamping [31]. All these different forces play a unique role in how the needle deflects, unfortunately it is incredibly difficult for a surgeon to predict the individual components of the reaction force, felt at the base of the needle. Knowledge of the individual force components would enable models to better predict the needle manipulations required to reach a subcutaneous target. A non-physics-based approach presented in [32] addressed the issue of estimating unobservable parameters to predict the reaction forces at the base of a needle during insertion. This approach used joint state-parameter extended Kalman filters to perform an online estimate of the LuGre model parameters. Authors of [33] explored how to account for friction during needle-based interventions. They noted how classical or static models are incapable of capturing the complex, highly non-linear dynamics of friction. They presented their finding using a distributed version of the LuGre model to characterize friction components of reaction forces while also predicting the cutting force. While [32] and [33] showed impressive characterization of different components of reaction forces, these models offered no way of predicting future trajectories.

1.3.3 Machine Learning in Needle-based Interventions

A recurring limitation of much of the work to-date can be attributed to modeling needle deflection using conventional physics and mechanics-based methods. Much of the complex needle-tissue interaction goes unmodelled. Many areas of research have seen an increased focus on techniques using ML. In [34] it was demonstrated how long-short term memory (LSTM) encoder networks could successfully predict the future trajectory of pedestrians.
and vehicles from a large data set. While the application of pedestrian movement prediction is very different than needle trajectory modeling, the efficacy of trajectory prediction is encouraging as pedestrians’ movements are highly coupled and influenced by many factors, similar to a needle cutting through tissue. Machine learning algorithms have gained some traction over the past couple years in the domain of robotic needle-based interventions. One popular subset of this research is the use of ML algorithms to analyze raw data from imaging equipment to locate tumors. Other groups have investigated using ML to facilitate path planning. The authors of [35] demonstrated that reinforcement learning (RL), which is a subset of ML, could be used to plan an optimal needle trajectory in the presence of anatomical obstacles using their novel Universal Distributional Q-learning algorithm.

1.3.4 Current Gap in Robotic Needle-based Interventions

While much work has been done in needle-tissue modeling, there still exists a gap in research for adaptive methods that predict the future state/trajectory of a needle in response to manipulations of the needle at its current location. The ability of a system to predict future states in response to current actions is essential for the planning and execution of needle control. Much of the work presented in the literature review presents methods of predicting current states that are not directly observable. These methods however fall short when tasked with predicting future states/trajectories of a needle. Certain work reviewed, such as the kinematic models in [25] can be extrapolated to make predictions, however those models do not adapt as the needle-based intervention evolves. Works such as [29], [32] presented adaptive models but only predicted currents states. Advances in hardware capabilities over the past decade have facilitated the use of machine learning (ML) algorithms for system modeling and control. Such algorithms are capable of learning highly non-linear dynamics of complex systems from real data. The improved hardware and continually growing number of powerful ML software libraries has made it possible to explore using ML techniques to improve the quality of care for those undergoing subcutaneous needle-based interventions. It would be wise to explore using ML algorithms to fill gaps which exist in current needle-tissue models.
1.4 Contributions

This thesis focuses on strategies to predict future state/trajectories of needle-based interventions in response to all possible manipulations that can be made at the current state/location of the needle. Building on that, the trajectory prediction methods presented are used within an optimal control framework to determine the best candidate of needle manipulations that can be performed to steer a needle to its intended target. The application of interest for this thesis is robotic assisted brachytherapy, although much of the work could be applied to biopsies as well. Rather than focusing on developing an analytical model for needle-tissue interaction, where errors are known to exist, this thesis explores using the shape of the needle to predict the future trajectory. This thesis also explores using modern machine learning techniques to simultaneously path plan and control needle manipulation.

Chapter 2 outlines novel state estimation and control algorithms that are used to continuously predict an optimal control strategy to steer a brachytherapy needle to its intended target. These algorithms make no assumptions on the needle-tissue interaction, instead adapting in real-time as the needle shape changes. Chapter 3 presents the experimental setup used to validate the proposed state estimation and control schemes presented in Chapter 3. Experimental results are presented and discussed. In Chapter 4, an end-to-end machine learning method that uses LSTM neural networks to design a low-level controller in the velocity domain while simultaneously planning a trajectory in the position domain is presented. Lastly, contributions are summarized and suggestions for future work/improvements are given in Chapter 5.
Chapter 2

2 State Estimation and Controller Design

This chapter presents different state estimation and control algorithms that are used to predict the future trajectory of a needle based on observations made throughout the evolution of a needle-based intervention. Two different approaches for predicting the future trajectory of a needle-tip during a needle-based intervention are explored. Both strategies look to exploit a hypothesis that the second derivative of needle-tip motion is related to the orientation of the bevel tip, and that rotating the needle to adjust the orientation of the bevel tip changes the vector describing the second derivative, and thus can be exploited to steer the needle. The first approach uses derivative estimates of the needle-tip position with respect to time to extrapolate the future trajectory of the needle. This approach uses a novel Adaptive LSTM Savitz-Golay (LSTM-SG) filter to estimate needle-tip derivative information from position measurements. The second approach uses the shape of the needle to predict the future trajectory. A Bezier curve (BC) is used to represent the shape of the needle. The BC method allows for a non-constant second derivative of the predicted trajectory. It is expected that this property is particularly useful when inserting the needle through non-homogenous material such as tissue, where a constant second derivative is unlikely. Both strategies can be used within an optimization framework to explore potential manipulation strategies and trajectory candidates. A state machine is used to control the logic which oversees the evolution of the insertion. The state machine will only act on the optimal needle manipulation candidate should the proposed strategy meet some minimum accuracy requirements that are intended to be determined preoperatively.

During development of the Bezier curve method, it was discovered that the tangent of the curve could be used to predict a reasonable estimate of the needle tip velocity. This estimate was used as an input to a second-order Kalman filter to improve position tracking of electromagnetic position sensors in the presence of electromagnetic interference. The remainder of this chapter assumes that a measurement of the needle-tip position is available, and that noise is present in the position measurements. A brief review of different state estimation strategies is given below. Following the review, both trajectory prediction
methods are explained. The way in which the second derivative vector is manipulated so that all possible trajectories can be explored is then presented for each method. Finally, a description on how the trajectory prediction methods are used within an optimization framework is given.

2.1 State Estimation Review

2.1.1 Kalman Filter Review

A Kalman filter [36] is a popular recursive filtering approach that combines system models with measurements, possibly even redundant measurements, to obtain an optimal estimate of a system’s state. The basic Kalman filter assumes that measurement noise and system process noise are Gaussian in nature while also assuming the system is linear and time-invariant (LTI). With this assumption, a Kalman filter produces an optimal estimate of the system state. Should the measurement and process noise not exactly follow a Gaussian distribution, Kalman filters will often still produce accurate estimates of system states. Kalman filters operate in a recursive “update” and “predict” cycle. At each time step, an update is performed on the state of the system model. A prediction of the system state and uncertainty for the next time step is then made. This cycle repeats at each time step. The model of the system is used when making predictions, while characteristics about measurements and system process noise are used for the update steps. The recursive Kalman filter algorithm is illustrated in Figure 2.1.

![Kalman Filter Algorithm](image)

Figure 2.1 - Kalman Filter Algorithm
2.1.1.1 State Extrapolation Equation

The state extrapolation equation is responsible for using a model of the system to predict the state of the system at the next time step. The state extrapolation equation is given in (2.1). The estimated state of the system is denoted by \( \hat{x} \) whereas the unknown true state is denoted by \( x \). The first index in the subscript for the estimated state represents the time step which the estimate is for, while the second index represents the time step when that prediction was calculated. \( F \) and \( G \) are the state transition and control matrices respectively. \( F \) maps the state of the system at the current time step to the state at the next time step while the system is not excited by external inputs. \( G \) maps the system inputs to incremental changes in the system state. \( w_n \) is the Gaussian process noise. While this quantity is not measured, it is included in the equation so that the optimal value of the Kalman gain can be derived.

\[
\hat{x}_{n+1,n} = F\hat{x}_{n,n} + Gu_n + w_n
\]  
(2.1)

The state transition matrix can be obtained directly from a matrix equation that describes the evolution of the states over time if such equations are available. Alternatively, If the system under consideration is given in the form \( \dot{x} = Ax \), the state transition matrix can be derived by calculating the matrix exponential of \( A \). The calculations for the state transition matrix and control matrix are given explicitly in (2.2) and (2.3) where \( \Delta t \) is the duration of the time step that the Kalman filter uses:

\[
F = e^{A\Delta t}
\]  
(2.2)

\[
G = \int_{0}^{\Delta t} e^{A\Delta t} B
\]  
(2.3)

2.1.1.2 Covariance Extrapolation Equation

The covariance extrapolation equation is shown in (2.4) below where the covariance matrix, \( P \), represents the uncertainty of the state estimate.
\[ P_{n+1,n} = FP_{n,n}F^T + Q_n \] (2.4)

It will be explained later how the recursive algorithm of the Kalman filter is designed to minimize the uncertainty of the state estimate. The uncertainty of the system states decreases over time as the algorithm sees more data points, thus having the opportunity to better tune the parameters. The predicted covariance matrix is used at the next time step to update both the Kalman gain and the covariance matrix itself.

\[ P_{n+1,n} = FP_{n,n}F^T + Q_n \]

### 2.1.1.3 Kalman Gain Update Equation

The Kalman gain, given in (2.5), is chosen such that it minimizes the trace of the covariance matrix.

\[ K_n = P_{n,n-1}H^T(HP_{n,n-1}H^T + R)^{-1} \] (2.5)

Minimizing the trace of the covariance matrix is analogous to minimizing the cumulative error of the state estimates. The exact expression for the Kalman gain is found by differentiating the covariance matrix with respect to \( K \), setting the expression equal to zero and then solving for \( K \).

\[ K_n = P_{n,n-1}H^T(HP_{n,n-1}H^T + R)^{-1} \]

### 2.1.1.4 State Update Equation

The state update equation is as follows:

\[ X_{n,n} = X_{n,n-1} + K_n(Z_n - HX_{n,n-1}) \] (2.6)

The equation uses the prediction of the current state, calculated at the previous time step, along with the measurement at the current time step, to make an optimal estimate of the system state. The Kalman gain is used to optimally weigh the predictions vs measurements when updating the current state estimate.

\[ X_{n,n} = X_{n,n-1} + K_n(Z_n - HX_{n,n-1}) \]
2.1.1.5 Covariance Update Equation

The covariance update equation is given by:

\[
P_{n,n} = (I - K_n H) P_{n,n-1} (I - K_n H)^T + K_n R K_n^T
\]  

(2.7)

A new variable seen in this equation is the measurement uncertainty matrix, \( R \). This matrix describes the expected noise from sensor measurements. Often, this matrix is a diagonal matrix where the values along the diagonal represent the expected variance for each measurement. This matrix can however have off-diagonal terms if the relationship between the error of different measurements is correlated and known.

\[
P_{n,n} = (I - K_n H) P_{n,n-1} (I - K_n H)^T + K_n R K_n^T
\]

2.1.2 Savitzky–Golay Filter Review

Savitzky-Golay (SG) filtering [37] is a technique often used to smooth noisy data. This technique is typically performed off-line as the algorithm is non-causal. The principle of the SG filter is that a curve is locally fitted to a small number of consecutive data points. There are \( N \) data points used to determine a curve with the best fit. The algorithm populates a buffer of length \( N \), where \( N \) is an odd positive number, with consecutive data points. The degree of the SG filter, \( M \), is equal to the order of the curve used to fit the data. The algorithm assumes that the data points are equally spaced in increments of 1 along the input domain. Linear least squares regression is performed to determine a curve with the best fit. Recall the form of equations for linear least squares is \( Ax=b \). The reason the SG filter requires data points equally spaced by increments of 1 along the input domain is that the \( A \) matrix can then take on the form of a Vandermonde matrix and thus the coefficients are easily calculated. The general form of a Vandermonde matrix is given by:
Because SG filters approximate data with smooth curves, the derivative(s) of signals can also be estimated using the derivative(s) of the curve. A benefit when using an SG filter is that the derivative estimates come with very little additional computational cost. This is best demonstrated with an example. Assume a set of 5 consecutive data points is given, \( Y \in \mathbb{R}^5 \), recorded at timesteps \( n = \{1, 2, 3, 4, 5\} \). Because the Vandermonde matrix is zero-centered, the timestep indices are shifted so that the data point in the middle of the buffer is the zero\(^{th}\) time step. The new timestep indices would be \( n' = \{-2, -1, 0, 1, 2\} \). A numeric example of the Vandermonde Matrix for a 2\(^{nd}\) order filter with a buffer of length 5 is given in (2.9). The coefficients of the second order curve can be calculated according to (2.10). Because the Vandermonde matrix is zero-centered and the filter is evaluated at the middle timestep, 0, many terms in the equation of the curve are cancelled by a multiplication of zero. Equation (2.11) highlights the only non-zero terms in red. The estimate of the signal and the first two derivatives are given by \( c_0 \), \( c_1 \), and \( 2 \times c_2 \) respectively. This shows how the derivative estimates come with little additional computational cost.
\[ V_5^2 = \begin{bmatrix} 1 & -2^1 & -2^2 \\ 1 & -1^1 & -1^2 \\ 1 & 0^1 & 0^2 \\ 1 & 1^1 & 1^2 \\ 1 & 2^1 & 2^2 \end{bmatrix} = \begin{bmatrix} 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix} \] (2.9)

\[
\begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = (V^T V)^{-1} V^T Y \] (2.10)

\[
f(t) = c_0 + c_1 \times t + c_2 \times t^2
\]

\[
\dot{f}(t) = c_1 + 2 \times c_2 \times t
\]

\[
\ddot{f}(t) = 2 \times c_2
\] (2.11)

While the basic SG filter is computationally efficient when working with data offline, there are two fundamental issues for implementing the filter in real-time, namely: (a) the filter is non casual and as a result, introduces lag in the state estimate, and (b) the incoming signals do not always arrive in increments of exactly one. Some modifications need to be made to make this filter appropriate for a real-time application. The modifications to the filter are presented later.

### 2.1.3 Bezier Curve Review

Motivated by the work in [38], where the authors showed that a second-order Bezier curve (BC) could be accurately fitted to a needle that has undergone a single rotation of 180 degrees during an insertion, it was decided that the use of a higher order BC would be explored for modelling needle shape. A third-order curve was chosen because a curve of this order does not constrain the shape of the needle to planar deflection while also not requiring the second derivative to be constant. A brief overview of BC and how its parameters are determined such that it fits a needle trajectory is given, followed by a method for using a higher-order BC in combination with a standard Kalman filter to filter noisy measurements. An important point to note is that this method makes no assumptions on the shape of the needle, instead it allows an optimization method to determine BC parameters such that the curve accurately describes the shape of the needle at each time step.
Bezier curves are different in comparison to other common types of curves in that instead of determining equation coefficients such that the curve coincides with specific points, a BC uses control points to form what is known as the control polygon. The control polygon is formed by drawing a straight line between the start/end points and the control points in the appropriate order, i.e., the start point is connected to the first control point, the control point to the second control point, and the last control point to the end point. These points can be thought of as “pulling” the curve towards them. As a result, BCs are capable of fitting nonlinear curves with unique shapes very accurately without making assumptions about the shape of the curve. A BC is a parametric curve where any point along the curve can be described by the curve parameter $t$. The equation of a third-order BC is given by (2.12):

$$BC(t) = (1 - t)^3 \cdot P_0 + 3 \cdot (1 - t)^2 \cdot t \cdot P_1 + 3 \cdot (1 - t) \cdot t^2 \cdot P_2 + t^3 \cdot P_3$$

(2.12)

The curve is drawn by plotting the points along the curve for $t \in [0,1]$. A BC of order $N$ is the combination of two BCs of order $N-1$. Each of the subsequent curves are also the combination of two curves of one less degree. The points along the BC can be traced out by linearly interpolating between the two curves which are combined to form the single higher order BC. This is best illustrated by Figure 2.2 where the linearly interpolated intermediate points are shown for a cubic curve. The green lines are linear interpolations between the start/end and control points. The blue line is an interpolation between the first pair of interpolation lines (blue). Finally, the Bezier curve is traced out in red. As the curve parameter $t$ varies from 0 to 1, the start/end points of the green and blue curves advance to the next control point in the control polygon. The BC is traced out by tracing all points along the blue curve for values for $t \in [0,1]$, as the green and blue curve simultaneously move. A third-order BC is described by a start/end pair of points and 2 control points.
To describe the shape of a needle with a BC, an optimization problem is solved at each timestep to fit a third-order BC to the trajectory of the needle-tip measurements observed up until that moment in time. The trajectory of the needle is stored in a buffer that is populated with position measurements of the needle tip beginning when the needle insertion begins. Recall that the reason for fitting the curve is so that predictions can be made about the future position of the needle-tip, a higher-order BC has a high capacity for accurately fitting data; however, this may result in “over fitting” such that the curve has erratic behaviour for points along the curve where the curve parameter is outside the domain \([0,1]\). This behaviour can be avoided by applying appropriate constraints to the location of the control points. The method of fitting a BC to the needle-tip trajectory is described next.

An optimization problem is set up to find the location of the control points of the BC such that the BC coincides with the measurements in the buffer as closely as possible. To reduce the complexity of the optimization problem, a group of \(N\) evenly spaced points within the buffer can be used to fit the curve instead of all points. Let \(P \in \mathbb{R}^{3 \times N}\) be an array populated with the measurements in the position buffer. The starting point of the BC, \(P_0\), is given by the first column of \(P\). In the absence of measurement noise, the end point \(P_3\), would be given by the final column of \(P\); however explicitly requiring \(P_3\) to coincide with the most recent position measurement may make the tangent of the curve susceptible to measurement noise. Instead, it is better to require that \(P_3\) be in the neighborhood of the most recent measurement and allow the optimizer to determine the optimal location of \(P_3\) such that the described cost function is minimized. Describing the Cartesian position of the final curve point adds three parameters to the optimization problem. The search space of
the optimization problem is thus a $9 + N$ dimensional space where the constant, 9, is the result of 3 parameters for each of the two control point locations $P1$ and $P2$, plus 3 parameters for the final curve point $P3$. The additional $N$ dimensions are the curve parameters $t_n, \{ t_n \in \mathbb{R}^N, n \in \mathbb{N} \cap [1, N] \}$ which describe the points along the BC that are closest to each sampled point in $P$. $N$ is chosen such that $N = \min(\# TrajectoryPoints, N_{max})$ where $N_{max}$ is a hyperparameter that can be tuned. The final vector of parameters that the optimizer solves for is given by (2.13):

$$\theta = [P_{1x}, P_{1y}, P_{1z}, P_{2x}, P_{2y}, P_{2z}, P_{3x}, P_{3y}, P_{3z}, t_1, t_2, \ldots, t_{N-1}, t_N]$$  \hspace{1cm} (2.13)

The initial conditions (ICs) for the parameter vector must be provided to the solver. The ICs for the control points are calculated by approximating the control point locations to be located along the needle axis, in increments of one-third the length of the trajectory in the buffer. The needle axis vector can be calculated using the rotation matrix, $R$, describing the orientation of the end-effector of the robotic manipulator performing the procedure. The IC for the control points is thus calculated according to (2.14):

$$
\begin{bmatrix}
P_{1xIC} \\
P_{1yIC} \\
P_{1zIC}
\end{bmatrix} = P0 + R \begin{bmatrix}
0 \\
\frac{L}{3} \\
\frac{L}{2}
\end{bmatrix},
\begin{bmatrix}
P_{2xIC} \\
P_{2yIC} \\
P_{2zIC}
\end{bmatrix} = P0 + R \begin{bmatrix}
0 \\
\frac{3L}{2} \\
\frac{L}{2}
\end{bmatrix},
\begin{bmatrix}
P_{3xIC} \\
P_{3yIC} \\
P_{3zIC}
\end{bmatrix} = P(end)$$  \hspace{1cm} (2.14)

The IC for the final curve point is given by the final column in $P$. The ICs for the curve parameters of closest points along the curve to sampled points are given by (2.15):

$$\theta_{n+9} = \frac{L * n}{N} \quad | \quad n \in \mathbb{N} \cap [1, N]$$  \hspace{1cm} (2.15)

It is assumed that the evenly spaced samples in the trajectory buffer will have curve parameters, $t_n$, approximately evenly spaced over the domain (0,1). $L$ is the depth of the needle tip in the tissue.
\[
\begin{bmatrix}
P_{1xIC} \\
P_{1yIC} \\
P_{1zIC}
\end{bmatrix} = P_0 + R * \begin{bmatrix} 0 \\ 0 \\ \frac{L}{3} \end{bmatrix}, \quad \begin{bmatrix}
P_{2xIC} \\
P_{2yIC} \\
P_{2zIC}
\end{bmatrix} = P_0 + R * \begin{bmatrix} 0 \\ 0 \\ \frac{3L}{2} \end{bmatrix}, \quad \begin{bmatrix}
P_{3xIC} \\
P_{3yIC} \\
P_{3zIC}
\end{bmatrix} = P(\text{end})
\]

\[\theta_{n+9} = \frac{L * n}{N} \quad n \in \mathbb{N} \cap [1, N]\]

To ensure the BC shape is not erratic outside the domain \(t \in [0,1]\), constraints must be placed on the solution of the optimization problem. Constraints must be enforced such that the vector formed by \(P_0\) and the control point \(P_X\), where \(P_X\) is the control point X, onto the Z-axis of \(R\), is greater than the projection of the vector \(\overline{P_0P_{X-1}}\) and smaller than the projection of \(\overline{P_0P_{X+1}}\). A constraint must also be enforced to ensure that the projection of the control points onto the Z-axis of \(R\) is positive. The control point constraints are given in (2.16).

\[
\begin{align*}
0 < \text{Proj}_Z \overline{P_0P_1} \\
\overline{P_0P_1} < \text{Proj}_Z \overline{P_0P_2} \\
\overline{P_0P_2} < \text{Proj}_Z \overline{P_0P_3}
\end{align*}
\]

(2.16)

\(\text{Where } Z := \text{ Needle Axis Vector}\)

There must also be a constraint placed on the total length of the curve to prevent the optimizer from finding a solution for the parameters of \(P_3\) that does not make sense geometrically, i.e., the resultant total curve length is not equal to the distance that the base of the needle has travelled. This constraint implicitly assumes that the needle does not undergo axial deformation during insertion. The constraint will enforce that the length of the curve is equal to the linear distance the end-effector of the robot has moved since the needle-tip positions began populating the array \(P\). Let \(s\) be the distance that the base of the needle has traveled and \(v\) be the linear velocity of the needle base during insertion. This constraint is given in (2.17).
\[ s = \int_{0}^{1} \sqrt{\left(\frac{dB(t)x}{dt}\right)^2 + \left(\frac{dB(t)y}{dt}\right)^2 + \left(\frac{dB(t)z}{dt}\right)^2} 
\]

where \( s := \int vdt \)  

(2.17)

The components of the norm in (2.17) are the x, y, and z derivatives of the Bezier curve with respect to the curve parameter \( t \). The integral is computed numerically using the fourth-order Runge-Kutta integration method. The cost function used to fit the BC to the buffer is given by (2.18).

\[ J = \sum_{1}^{N} \| P_n - B(t_n) \|^2 \]

where \( P_n = n^{th} \) column of \( P \)  

(2.18)

### 2.2 State Estimation Implementation

#### 2.2.1 Velocity Estimation Using Bezier Curve Tangent

While the primary role of the fitted BC is in its use for trajectory prediction, the tangent of the BC can be used for velocity measurement in a Kalman filter (KF) to improve position tracking. The tip of a needle being inserted into tissue follows a nonlinear path; however, with a sufficient sample rate, the motion of the needle-tip over the duration of a time step can be approximated as linear. A constant velocity KF is employed to filter noisy position measurements of the needle-tip. The state of the KF is given by:

\[ X_n = \begin{bmatrix} x_n \\ \dot{x}_n \\ y_n \\ \dot{y}_n \\ z_n \\ \dot{z}_n \end{bmatrix} \]

(2.19)
The state transition matrix is shown in.

\[
F = \begin{bmatrix}
1 & \Delta T & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & \Delta T & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}
\] (2.20)

Recall that noisy position measurements of the needle-tip are available. There is however no direct measurement of the needle tip velocity. Evaluating the tangent of the BC at the needle-tip position, scaled by the appropriate linear velocity, provides an estimate of the needle-tip velocity vector. There is thus a measurement available for each state of the system, the measurement equation is given in.

\[
\begin{bmatrix}
x_n \\
y_n \\
z_n
\end{bmatrix} = H \begin{bmatrix}
x_n \\
y_n \\
z_n
\end{bmatrix} = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\] (2.21)

The system is assumed to have variance in each Cartesian velocity, the process noise matrix is calculated according to (2.22).

\[
Q = F \begin{bmatrix}
Q_a & 0 & 0 \\
0 & Q_a & 0 \\
0 & 0 & Q_a
\end{bmatrix} F^T, \quad \text{where } Q_a = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & \sigma^2_x
\end{bmatrix}
\] (2.22)

### 2.2.2 Adaptive LSTM Savitzky-Golay Filter

The algorithms presented later in this chapter require accurate derivative estimates from noisy position data. An objective of this research was to explore using modern machine learning techniques to improve the quality of needle-based interventions. The trajectory of the research, presented in this thesis, highlighted an opportunity to use ML to improve state estimation for nonlinear systems. This section first provides details on how the SG filter can be modified so that it can operate in a non-causal fashion. A shortcoming of the SG-filter is demonstrated with a brief example before the proposed solution is presented.
Simulation results are presented to demonstrate superior performance of the proposed method when compared to other state estimation methods.

Recall that the SG filter is designed to be a non-casual algorithm with measurements incoming at a fixed sample rate of 1 measurement per 1 unit of the input domain. For the purposes of this algorithm, the input domain is time. If the sample rate of measurements is not 1 Hz, the $A$ matrix in the equation $Ax = b$ can no longer assume the form of a Vandermonde matrix. Let $a \in \mathbb{R}^N$ be a vector that is populated with the time stamps from the $N$ measurements in the filter buffer. Let $\hat{a} \in \mathbb{R}^N$ be a vector that is zero centred at the middle value of $a$. The $A$ matrix for the SG filter can be computed by populating each column with an element-wise power of $\hat{a}$ where the power for column $i$ is $i - 1$. Equations (2.23) and (2.24) outline the steps for computing $A$.

$$a = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_{(\frac{N}{2})-1} \\ t_{N-1} \\ t_N \end{bmatrix}, \quad \hat{a} = \begin{bmatrix} t_1 - t_{(\frac{N}{2})-1} \\ t_2 - t_{(\frac{N}{2})-1} \\ \vdots \\ 0 \\ \vdots \\ t_{N-1} - t_{(\frac{N}{2})-1} \\ t_N - t_{(\frac{N}{2})-1} \end{bmatrix} \quad (2.23)$$

$$A = [\hat{a}^0 \ \hat{a}^1 \ \hat{a}^2] \quad (2.24)$$

If the incoming sample rate of measurements is constant, the $A$ matrix needs to be calculated just once. The lag introduced by evaluating the position and derivatives in the center of the filter buffer can be eliminated by evaluating the position and the derivatives at the most recent entry to the buffer. This will make the estimates more susceptible to noise; however, a strategy for mitigating the impact of the noise is addressed in the following section. The state estimates from the most recent time step can be calculated according to (2.11), however using all terms, not just the terms highlighted in red. Note that $t$ in (2.11) is given by $t = t_N - t_{(\frac{N}{2})-1}$, which may or may not be constant depending on the consistency of the sample rate.
When designing an SG-filter, there is a trade-off to be made between lag and accuracy. A shorter filter length will typically reduce the lag of the filter but will come at the cost of greater noise. Consider Figure 2.3 where a piecewise function is plotted by the solid red line. Noise is added to the signal before it is sampled at 10 Hz. This is repeated 10 times. The noise is different for each of the sampled signals, thus there are ten different, noisy discrete signals. The ten discrete signals are plotted in Figure 2.3 using a scatter plot with circular markers.

![True Position and Noisy Position Measurements](image)

**Figure 2.3: Piecewise function with non-continuous second derivative**

At $t = 2.7s$, the function abruptly changes from a pure quadratic function to the sum of the previous quadratic and a larger, negative quadratic. The dotted red line shows the function from the first segment of the piecewise function if it were to continue past $t = 2.7s$. In the case of an abrupt change in dynamics, a shorter filter length would reduce the number of time steps required for the filter’s derivative estimates to reflect the change in system dynamics. In contrast, during a period of steady state continuous dynamics, a longer filter length will typically increase the accuracy because the noise from each measurement has less influence on the overall fit of the curve used in the SG-filter. Figure 2.4 illustrates how the filter length that produces the most accurate derivative estimates changes with time.
Figure 2.4: Illustration of shorter filter length producing better derivative estimates when the signal of interest does not follow steady state dynamics

In the figure, the filter estimation error is plotted against filter length for three different time steps from the function in Figure 2.3. The optimal filter length for each time step is shown with an ‘X’. It is observed that the optimal filter length begins to decrease immediately following the change in the piecewise function. This motivates finding a strategy that can adaptively change the filter length during estimation. The challenge is finding a method that can predict an optimal filter length in real-time. While the adaptive filtering strategy described next was created for the application of this thesis, it is expected that this technique could be applied to a broad range of domains, provided there is sufficient training data. To facilitate an adaptively changing filter length, machine learning was employed. The specific type of machine learning that was used for this application was a class of recurrent neural networks called Long Short-Term Memory (LSTM) networks. These types of networks are a great candidate for this application because they have memory and can learn abstract temporal patterns such as when a transient in the dynamics of measurements has taken place. The LSTM network will determine if a measurement transient warrants a shortened filter length to gain a more immediate response in state estimation, or if the estimation would benefit from a longer filter length. For this
application, the input to the network is the sequence of position measurements which are to be filtered, while the output is a prediction of the optimal filter length for each time step in that input sequence. The optimal filter length is determined during the data pre-processing step.

A double integrator system is modelled in simulation to generate the data used to train the LSTM network. All three states of the system (position, velocity, acceleration) were recorded for the duration of each simulated trajectory. The input to the first integrator (acceleration) was modeled as a series of step functions that are intended to be analogous to abrupt changes in tissue stiffness. Noise was added to the position measurement and recorded as a fourth signal. The simulation clock was also recorded. 100,000 simulations were run to generate a dataset to train the LSTM network.

The input sequences used for training were the noisy position measurement sequences. Through many episodes of training, it was determined that concatenating the same sequence with itself 5 times, each of the sequences offset by 2 time steps, resulted in the best training outcomes. The offset sequences were zero-padded at the beginning. The network targets were the optimal filter lengths for each timestep. A search was performed to determine the optimal filter length for each time step of each sequence. For each such time step, an SG filter was fitted to the N most recent position measurements where N is an odd integer in the range $[3,60]$. Note that the first two time steps of each sequence are excluded from the search due to an insufficient number of samples for a second-order filter. The errors in the first and second derivative estimates are stored in memory for each value of N. After calculating the errors for all filter lengths at a given time step, the stored error values were examined to determine the filter length that resulted in the most accurate estimate. This process was repeated for all time steps of all training sequences. To shorten the time required to find network targets, the process was parallelized using the `parfor` function in MATLAB. The derivatives of the noisy position data were calculated using an SG-filter of length $N$ for a range of filter lengths where $N$ is an odd integer ranging between 1 and 60. By comparing the derivative estimates, for each filter size, with the true known values, the optimal filter length could be determined. This was performed for each time step of each simulated sequence.
With the inputs and targets calculated, the dataset was split into training, validation and testing data using an 80/15/5 split. All input sequences were shifted to have a zero mean and normalized to have a standard deviation of 1. The target optimal filter lengths were zero-centered and normalized for training. A network was constructed according to . This network had an LSTM layer containing 32 cells, followed by a fully connected (FC) layer with 16 neurons.

![LSTM Network Diagram](image)

**Figure 2.5 – Adaptive filter length LSTM Network**

The network was trained using truncated back propagation for 140 epochs. A dropout layer was used between the 16 neuron FC layer and the output layer to reduce overfitting. Figure 2.6 plots the prediction and true values of optimal filter size for a random sequence against time.
**Figure 2.6 – Optimal filter length prediction with LSTM network**

Figure 2.7 plots the RMSE and SD of the optimal filter length prediction error against the number of samples used to calculate those two metrics. Both the RMSE and SD converge to steady-state values as the number of samples used to calculate RMSE and SD increase.

**Figure 2.7: Convergence of LSTM predicted RMSE and SD of RMSE.**
Figure 2.8 illustrates the comparison of the SG-filter where the filter length is calculated with three methods, namely (a) fixed length, (b) optimal length found using a grid search, and (c) predicted optimal length using an LSTM network. Two lines are plotted to compare the performance of velocity and acceleration estimation for all three filters. These lines are estimation errors from the ratios of (c)/(a) and (b)/(a). The ratios are plotted against the number of different sequences used to calculate the ratios.

**Figure 2.8: Convergence of the ratios comparing adaptive and optimal filter lengths with fixed filter length.**

Figure 2.8 shows the convergence of these ratios. The ratios for the filter length found using the LSTM network, shown in the upper plot, both converge to values less than one, indicating that the SG-filter with the adaptive filter length outperforms the fixed-length filter. The ratios of the filter that used the true optimal length are superior when compared to the LSTM predicted optimal lengths. These results are included to demonstrate the ceiling for an adaptive LSTM SG-filter performance. It is expected that with a larger dataset, the performance of the LSTM SG-filter would approach that of the filter which uses the true optimal filter lengths.

An important objective of the research was to demonstrate the superior performance of an adaptive LSTM SG-filter when compared to other common state estimation techniques.
The LSTM SG filter was compared to a three-state KF, with the state vector comprised of position, velocity, and acceleration. The system is assumed to have a constant second derivative with a specified variance. The state transition equation is given in

\[
\begin{bmatrix}
  x_{n+1} \\
  \dot{x}_{n+1} \\
  \ddot{x}_{n+1}
\end{bmatrix} =
\begin{bmatrix}
  1 & \Delta t & \frac{1}{2} \Delta t^2 \\
  0 & 1 & \Delta t \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x_n \\
  \dot{x}_n \\
  \ddot{x}_n
\end{bmatrix}
\]

(2.25)

The process noise uncertainty matrix was computed under the assumption that there exists variance in the second derivative. Using the state transition matrix, the process noise matrix for the system was then calculated substituting the constant acceleration variance matrix, shown in (2.26), into (2.27). The observation matrix is shown in (2.28).

\[
Q_s = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & \sigma^2_{\ddot{x}}
\end{bmatrix}
\]

(2.26)

\[
Q = F Q_s F^T
\]

(2.27)

\[
H = \begin{bmatrix}
1 & 0 & 0
\end{bmatrix}
\]

(2.28)

The KF was tuned using the training dataset. The measurement and process noise variance that had the best performance were determined to be 0.005 and 0.2 respectively. The metric used for evaluation was the running RMSE of the first two derivatives over the duration of each sequence. Similar to Figure 2.8, the ratio of the LSTM SG-filter/Kalman filter error were plotted in Figure 2.9, against a number of different sequences used to calculate the ratio.
Figure 2.9: Convergence of the ratios comparing first and second derivative estimates of LSTM SG-Filter with Kalman Filter.

The figure illustrates the ratio of the error for both velocity and acceleration converging to values less than one. The first derivative saw close to 45% decrease in error while the second derivative experienced almost 60% decrease in error. The improved performance of the adaptive LSTM SG-filter is illustrated in Figure 2.10, where the acceleration estimates from both filters are plotted alongside the true value.
Figure 2.10: Comparison between adaptive LSTM SG filter and Kalman filter second derivative estimate

The LSTM network identifies when a transient has occurred at $t = 6s$ and responds by requesting an immediate shortening of the filter. The effect of an adaptive filter length enables the second derivative estimate to converge much faster when compared to the Kalman filter. Figure 2.10 also illustrates a shortcoming of the proposed filter, which is that the derivative estimates are very sporadic at the beginning when there is a limited number of measurements for fitting the curve. A solution for this shortcoming is to implement some safeguards such that the derivative estimates are not used until a sufficient number of measurements have populated the buffer.

This section was focused on demonstrating the performance of the proposed adaptive LSTM SG-Filter where true known signals are available in simulation. In Chapter 3, details are provided regarding the performance of the filters when used for the purpose of needle-based interventions.
2.3 Control Architecture

A control scheme was designed to continuously search over the possible future needle trajectories at each moment in time during the insertion. The potential trajectories differ due to varying amounts of rotation that could be performed at the needle tip’s current location. The proposed idea is that the second derivative of the needle-tip position, described with respect to the base of the needle, can be attributed to the orientation of the needle’s bevel tip. The control algorithm continuously searches for the optimal amount of rotation. Once an optimal needle trajectory candidate meets some minimum performance threshold, the higher-level logic state machine will give the command to the needle velocity controller to execute the proposed optimal rotation. The control architecture can be split into three main components, trajectory prediction, optimization, and high-level logic. More details on each section are given below.

2.3.1 Trajectory Prediction

Two methods are proposed for predicting the future trajectory of the needle tip. The first method uses the derivative estimates from the adaptive LSTM SG-Filter to extrapolate the current needle tip position in time. This method performs extrapolation in the coordinate system rigidly attached to the base of the needle. The extrapolated trajectory is then transformed into task space to determine its proximity with the target. The second method predicts the trajectory by evaluating the BC for values of the curve parameter, \( t > 1 \). This method is carried out in cartesian space with respect to the ground frame of a robotic manipulator. Both methods are designed such that future reachable trajectories can be explored as a function of rotation applied to the base of a needle at its current location.

2.3.1.1 Adaptive LSTM SG-Filter Trajectory Prediction

Note that unless otherwise specified, the needle-tip positions for the remainder of this method’s explanation are assumed to be described with respect to a frame rigidly attached to the base of the needle. Figure 2.11 illustrates the assigned frames.
The notation used is as follows: $X$ is a $3 \times 1$ vector containing the $x$, $y$ and $z$ components of the needle tip with respect to the base of the needle. Similarly, $\dot{X}$ and $\ddot{X}$ are $3 \times 1$ vectors containing the $x$, $y$ and $z$ components of the first and second derivatives respectively of the needle tip position with respect to the base of the needle. The hat symbol on a vector indicates that vector is an estimate, i.e., $\hat{X}$ is an estimate of the first derivative of $X$. Some vectors are shown with subscripts. Any vector with a subscript 0 implies that it is an initial condition, i.e., $X_0$ denoted the initial position in equations 2.30 and 2.32. The value of this initial position is given by the most recent position measurement. The vector $\hat{X}_\theta$ is the transformed estimate of the second derivative of the needle tip position, where the transformation is given by a rotation matrix, $R_z(\theta)$, representing a rotation of $\theta$ radian about the $z$-axis. The variable, $\Delta T$, in (2.30) and (2.32) is not a fixed discrete timestep; it is a parameter in the optimization problem that is solved to find an optimal trajectory using the SG-filter. The position of the needle tip, along with the first two derivatives are used to predict the future trajectory. It is proposed that a non-zero second derivative of needle position can be attributed to the force acting on the asymmetric needle-tip. Rotating the
needle changes the direction of the cutting force and thus changes the direction of the second derivative. The $x$ and $y$ positions of the needle tip with respect to the base (zero$^{th}$ frame in Figure 2.11), along with its first two derivatives are shown in (2.29). The equation used to extrapolate the needle position is given in (2.30). To extrapolate a predicted trajectory for a needle that has been rotated by $\theta$ rad, a rotation matrix is used to transform the current $x$ and $y$ components of the second derivative which is analogous to the base of the needle having undergone a rotation. Equation (2.31) shows the transformation of the second derivative estimate. The trajectory is then extrapolated using the rotated second derivatives while using the same initial conditions for velocity and position as in (2.30). The extrapolated rotated trajectory is given by (2.32).

### 2.3.1.2 Bezier Curve Integration Trajectory Prediction

The second trajectory prediction method predicts the future trajectory by evaluating the curve for a curve parameter $t > 1$. Unlike the first method which assumes the second derivative is constant throughout the remainder of the insertion, the predicted trajectory found using this method may have a time-varying second directive which prevents finding the predicted trajectory using closed-form solutions, requiring integration to be performed. As with the first method, the Cartesian components of the BC’s second derivative must be rotated about the needle axis, to explore the effect of rotating the base of the needle. Let $R$ represent the rotation matrix describing the orientation of the base of the needle with respect to the task space coordinate system. A rotation matrix, rotating about the z-axis, is used to rotate the BC second derivative vector about the needle axis. Let this rotation matrix be given by $R_z(\theta)$. The Cartesian components of the second derivative, that have been rotated about the needle axis are given in.

$$
\mathbf{\ddot{BC}}_\theta(t) = R^T R_z(\theta) R \mathbf{\ddot{BC}}(t)
$$

(2.33)

The predicted future trajectory of a needle that has been rotated at its current position can be found by integrating the system given by.
The seventh state is the length of the integrated curve and is included in the state vector so that a stopping condition for integration can be triggered once the trajectory has been extrapolated a sufficient amount to evaluate its proximity to the target location. A fourth-order Runge-Kutta method is used to perform integration. The initial conditions for integration are given in (2.35).

\[
X_0 = \begin{bmatrix}
x_{\text{sensor}} \\
\dot{BC}_x(1) \\
y_{\text{sensor}} \\
\dot{BC}_y(1) \\
z_{\text{sensor}} \\
\dot{BC}_z(1) \\
0
\end{bmatrix}
\]  

(2.35)

### 2.3.2 Trajectory Optimization

Both trajectory prediction methods predict the future trajectory of a needle that has undergone a rotation of \( \theta \) radians about the needle axis, at its current position. The closest point to the target location along a predicted trajectory must be determined to evaluate the expected proximity to the target location. In the case of the first method, numerical optimization is used to search for the combination of \( \Delta T \) and \( \theta \) that results in the point along a predicted trajectory which minimizes the distance to the target location. The cost function is thus a function of two variables. While \( \theta \) is the main variable required to generate an optimal trajectory, the point along the extrapolated trajectory that is closest to the target location can be found using the same numerical optimization instead of performing integration because the extrapolated trajectory has closed-form solutions. The Bezier curve approach requires that the system be integrated until the curve length, \( s \), is of a value greater than \( d + \delta \), because it is expected that the needle will not travel in a
perfectly straight path. The term $\delta$ is a hyperparameter that can be tuned, while $d$ is the straight-line distance from the needle-tip to the target location. A search is performed on the integration steps where $d < s < d + \delta$ to determine the predicted proximity of the needle to the target location. To determine an optimal value of $\theta$, for either method, a cost function (2.36) returns the squared Euclidean norm between the target and the closest point along predicted trajectory.

$$J(t) = \|\hat{X}(t) - Target\|^2$$

Sequential-quadratic programming is applied to iterate over possible values of $\theta$ to determine the optimal value. Figure 2.12 shows the block diagram for the adaptive LSTM SG trajectory prediction strategy.

![Block Diagram](image)

**Figure 2.12: Trajectory prediction block diagram using an LSTM SG filter**

A block diagram illustrating the architecture of the BC trajectory prediction method is shown in Figure 2.13.
2.3.3 High-Level Logic

A state machine is used to implement the logic that oversees the progression of the needle insertion. After start-up, the state machine awaits the command to trigger the alignment of the needle. Aligning the needle requires that the orientation of the needle be adjusted so that the needle axis coincides with the target location. Once the needle is aligned correctly, the state machine awaits confirmation from the operator to initiate the insertion. Once initiated, the state estimation and control algorithms begin. The robotic system steering the needle (details given in Chapter 3) will begin performing translational insertion with a constant velocity. While in the translational, constant velocity state, the trajectory optimization algorithm will continuously search for an optimal needle manipulation strategy. If an optimal trajectory candidate is found, where the expected proximity to the target location is within the predetermined acceptable threshold, the state machine will pause translational motion and trigger an angular rotation of the needle. The amount of needle rotation, $\theta$, is determined by the trajectory optimization algorithm. A PID controller is employed to perform the correct amount of rotation. Once the rotation has finished, the translational insertion will resume. The state machine will continue to monitor for new optimal trajectory candidates. Excess rotation is known to cause a damaging drilling effect on muscle tissue, so this algorithm seeks to determine a needle manipulation strategy that
requires as few rotations as possible. To avoid many small rotations after the initial rotation, with negligible improvement(s) in accuracy, another threshold is set specifying how much more accurate an optimal trajectory candidate must be when compared to the current trajectory to warrant another rotation. The state machine is described by Algorithm 2.1. The experimental results are given in Chapter 3.

Algorithm 2.1: Needle Insertion Logic State Machine
Chapter 3

3 Experimental Setup and Results

Experiments were performed to evaluate the performance of the proposed state estimation and control strategies described in Chapter 2. Experiments were carried out to evaluate three items. The first item was to evaluate how the Kalman filter, using the Bezier curve tangent, performed at filtering noisy electromagnetic position measurements. The second item to evaluate was how accurately both trajectory prediction methods were able to predict future trajectories for different amounts of rotation. Lastly, the accuracy was evaluated when using each of the proposed methods within an optimization framework to locate and steer the needle to a target location. A 6 DOF manipulator was chosen to manipulate the needle in all experiments. The experimental setup is described below, followed by details on each evaluation experiment. The experiments described in this chapter are to validate the approaches proposed in Chapter 2, in a scenario where a robotic system performs a procedure without human intervention. While such a scenario would not be used in a clinical setting due to ethical and safety concerns, a successful outcome of these experiments and others like it can help the medical community to one day consider performing such procedures entirely under the control of appropriate "intelligent" robotic systems.

3.1 Experimental Setup

3.1.1 Robotic Manipulator

A six degrees-of-freedom robotic manipulator, the Gen2, manufactured by Kinova was used to manipulate the brachytherapy needle. The manipulator communicates with the host computer at 100 Hz via a USB connection. There is a C++ API provided by the manufacturer which is used to send commands to the manipulator. The robotic arm can be operated in position, velocity, and torque control. The joint-velocity controller was chosen since the control algorithm specifies needle manipulations in the velocity domain. A custom Simulink library was developed to bridge the communication between the Simulink controller and the robotic manipulator. This library contains a custom block with a level 2 C++ S-Function. The custom library block facilitates sending position, velocity and torque
commands to the manipulator while also oversees safety concerns such as the manipulator exceeding its predefined safe workspace. A photo of the manipulator is shown in Figure 3.1.

Figure 3.1: Kinova Gen2 6 degrees-of-freedom Robotic Manipulator

The proposed control algorithms specify Cartesian velocities with respect to a frame rigidly connected to the base of the needle, while the Gen2 accepts joint velocities. The relationship between Cartesian velocities described with respect to the needle-base, and the Gen2 joint velocities has to be determined to calculate the joint velocities required to perform the desired needle motion. This relationship is found by calculating the Jacobian of the end-effector frame. The derivation of the Jacobian is performed by first using the transformations between each link to determine the transformation matrix relating the end-effector to the base frame, described in the base frame coordinate system.
\[
\mathbf{T}_{ee} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ 0 & 1 \end{bmatrix} = \mathbf{T}_1^0 \mathbf{T}_2^1 \mathbf{T}_3^2 \mathbf{T}_4^3 \mathbf{T}_5^4 \mathbf{T}_6^5 = \mathbf{T}_6^0, \text{ where } \mathbf{R} \in \mathbb{R}^{3 \times 3}, \mathbf{p} \in \mathbb{R}^3
\] (3.1)

The resultant transformation matrix can be divided into a position vector and a rotation matrix. Taking the partial derivative of the position vector with respect to each joint angle yields the linear velocity Jacobian.

\[
0\mathbf{J}_v = \begin{bmatrix}
\frac{\partial}{\partial \theta_1} p_x & \frac{\partial}{\partial \theta_2} p_x & \cdots & \frac{\partial}{\partial \theta_6} p_x \\
\frac{\partial}{\partial \theta_1} p_y & \frac{\partial}{\partial \theta_2} p_y & \cdots & \frac{\partial}{\partial \theta_6} p_y \\
\frac{\partial}{\partial \theta_1} p_z & \frac{\partial}{\partial \theta_2} p_z & \cdots & \frac{\partial}{\partial \theta_6} p_z 
\end{bmatrix}
\] (3.2)

The angular Jacobian is constructed by placing the Z-axis of the rotation matrix relating frame \(i\) to the base frame into the \(i^{th}\) column of the angular Jacobian.

\[
0\mathbf{J}_w = \begin{bmatrix}
0_{R_z}^1 & 0_{R_z}^2 & \cdots & 0_{R_z}^6 
\end{bmatrix}
\] (3.3)

where \(0_{R_z}^i\) := 3\(^{rd}\) column of Rotation matrix relating link \(i\) to base frame

The Jacobian can then be transformed such that it describes motion with respect to the end-effector by performing the transform described in (3.4).

\[
6\mathbf{J} = \begin{bmatrix}
0_{R^T}^6 & 0 \\
0 & 0_{R^T}
\end{bmatrix} 0\mathbf{J}
\] (3.4)

The joint velocities that satisfy the desired motion, described in the end-effector frame, can be calculated using the relationship in (3.5).

\[
\dot{\theta} = 6\mathbf{J}^{-1} 6\dot{x}
\] (3.5)

It should be noted that the calculation of joint velocities requires the inversion of the Jacobian. An inversion of the Jacobian will yield very large values when the manipulator is close to a kinematic singularity. To avoid such velocities, the damped least squares (DLS) method was used to calculate the required joint velocities in the presence of kinematic singularities. The damped least squares equation is given in equation (3.6).
\[
\dot{q} = (J^T J + \lambda^2 I)^{-1} J^T 6\dot{x}
\] (3.6)

\[
\lambda = \left(1 - \left(\frac{S_{\text{min}}}{\epsilon}\right)^2\right) \lambda_{\text{max}}
\] (3.7)

There are a few parameters that appear in the equations for damped least squares. The term \(\lambda_{\text{max}}\) in (3.7) is defined as the maximum damping coefficient, i.e., the damping coefficient that can be used to control deviation from the desired trajectory in the scenario that the manipulator is close to a singularity configuration. Damped least squares need only be used in such a situation. The metric chosen to evaluate how close the manipulator is to a singularity is the smallest singular value of the Jacobian, denoted by \(S_{\text{min}}\). Another parameter, \(\epsilon\), is the user defined threshold for how small \(S_{\text{min}}\) must be before DLS is employed. Simulations were performed to find a set of DLS parameters that work well for the expected trajectories. The parameter that was chosen for \(\lambda_{\text{max}}\) and \(\epsilon\) was 0.04.

### 3.1.2 Force Sensor

An ATI Gamma six degrees-of-freedom force/torque (F/T) sensor was used to measure/record reaction forces at the base of the needle during insertions. This device communicates with the host computer at 100Hz via a hardware specific PCI DAQ card. A custom Simulink block was designed to integrate the sensor with the state estimation and control algorithms. The raw F/T sensor voltages are converted to meaningful coordinate system measurements using calibration coefficients provided by the manufacturer. The sensor was attached to the end-effector of the robotic arm using a custom designed 3D printed attachment. The needle was then rigidly attached to the distal end of the sensor so that the reaction forces at the base of the needle could be recorded. The CAD model and 3D printed attachment can be seen in Figure 3.2 and Figure 3.3 respectively. There is a bias in the force sensor measurements when the needle is attached to the distal end that needs to be compensated for. Because the mass of the needle is approximately evenly distributed about the axis which the needle will rotate about (if need be), and the direction of that axis remains constant throughout the insertion, the force sensor bias can be corrected by zeroing
the measurement values prior to commencing insertion. Inertial and centrifugal terms were neglected because of the relatively low velocity and mass [39].

Figure 3.2: Exploded view of CAD model of the custom-designed end-effector and needle mount
3.1.3 NDI Aurora Electro-Magnetic (EM) Position Tracking System

The Aurora electro-magnetic position tracking system, manufactured by Northern Digital Instruments (NDI) [40], was employed to measure the needle tip position as it was inserted into soft tissue. EM trackers were selected as they directly measure position, eliminating the need for post-processing techniques to extract the position measurements from the data received at the host computer. Additionally, the sensors are extremely small and can be housed within the needle, making for highly accurate tracking. While the experimental setup simply places a sensor within a hollow needle, a more viable option for a clinical setting would be to use the 18G needle manufactured by NDI, which houses a stylet that is embedded with a 5DOF EM tracker [41]. The efficacy of placing an EM tracker within a stylet as opposed to the needle tip has been validated in the past for an application involving lung brachytherapy [42].

There are three main components to the Aurora tracking system, the field generator, system control unit (SCU), and the EM sensors. The SCU is connected to the host computer via a USB connection. Again, a custom Simulink library was created to communicate with the device. The tracker measurements were recorded at 40Hz. Measurements are read as a vector, from the manufacturer provided C++ API, containing the Cartesian position, and orientation described using quaternions. The field generator used was the table-top model, boosting a measurement cross section 420 x 600 mm in the
shape of an ellipse, with a height of 600 mm. The measurement volume is offset from the table-top surface by 120 mm. An Aurora 5 DOF sensor, measuring 0.85 mm in diameter and 11 mm in length, was placed at the tip of the brachytherapy needle. This sensor was held in place on the interior of the needle wall using an epoxy, the wire connecting the sensor to the SCU ran along the inside of the needle and out the base of the needle before connecting to the SCU. While the experimental setup presented in this chapter uses electromagnetic trackers to obtain measurements of the needle-tip and the target position, an ultrasound imaging system could be more practical in a clinical setting, and could be used to determine the coordinates of the target location and needle tip. Such measurements could be substituted for the EM tracker measurements in the proposed algorithms.

Figure 3.4: Aurora NDI tabletop field generator [43]

Figure 3.5: Aurora NDI System Control Unit (SCU)

3.1.4 Aligning Coordinate Systems of Gen2 and Aurora

All hardware components must be able to communicate with respect to some common coordinate system. It was decided to transform the measurements from the Aurora coordinate frame to that of the Gen2. The transformation could also have been performed
such that the Gen2 coordinates were transformed into the Aurora coordinate system. Aligning the coordinate systems accurately is very important as the needle deflection with respect to the base of the needle is on the order of 10 mm. So even a small misalignment would be detrimental to the performance of the algorithms. To collect measurements in each coordinate system, a second EM tracker was secured in place on the end-effector, at a location matching a frame location on the kinematic chain of the Gen2. The end effector was moved to various positions throughout the measurement volume of the field generator. Once in place, the position of the sensor was recorded using both the measurement data from the Aurora system, and the forward kinematics of the Gen2. This process was repeated for 15 different locations. A familiarity covariance matrix, $H$, was constructed using the collected position measurements and then decomposed using singular value decomposition (SVD) to determine a rotation matrix and transformation vector relating the two coordinate systems. The steps to determine the rotation matrix and transform vector are as follows. Let $X_{Aurora}$ and $X_{Mico} \in \mathbb{R}^{3 \times 15}$ represent matrices containing the Cartesian positions of the 15 measurements recorded in both the field generator and Gen2 coordinate systems respectively. The position measurements in each coordinate system need to be zero-centered according to and to construct the familiarity matrix.

\[
\text{centroid}_{X_{Aurora}} = \frac{1}{N} \sum_{i=1}^{N} X_i^{\text{Aurora}}
\]

where $X_i^{\text{Aurora}} = \text{ith column of } X_{Aurora}$

\[
\text{centroid}_{X_{Mico}} = \frac{1}{N} \sum_{i=1}^{N} X_i^{\text{Mico}}
\]

where $X_i^{\text{Mico}} = \text{ith column of } X_{Mico}$

The familiarity covariance matrix was found according to (3.10).
\[
H = (X_{\text{Aurora}} - \text{centroid}_{X_{\text{Aurora}}})(X_{\text{Mico}} - \text{centroid}_{X_{\text{Mico}}})
\]  \hspace{1cm} (3.10)

The rotation matrix and linear transformation vector, transforming field generator Cartesian measurements into Gen2 measurements can be calculated by using the \( U, S, \) and \( V \) matrices found using SVD as shown in (3.11), (3.12), (3.13).

\[
[U, S, V] = \text{SVD}(H)
\]  \hspace{1cm} (3.11)

\[
\begin{align*}
M_{\text{Aurora}}^* & = VU^T \\
M_{\text{Aurora}}^* & = \text{centroid}_{X_{\text{Mico}}} - M_{\text{Aurora}}^* \ast \text{centroid}_{X_{\text{Aurora}}}
\end{align*}
\]  \hspace{1cm} (3.13)

Recall that the method of using the familiarity matrix assumes the EM tracker on the end-effector is placed precisely at the location where the frame is specified on the kinematic chain. This assumption is not valid outside of simulation, discrepancies of just 0.5 mm between the true and expected sensor location can have a significant effect on the performance of the proposed control algorithms. To find a transformation between the Gen2 and field generator coordinate system that accounts for a misplacement of the end-effector EM tracker, an optimization problem is setup that searches to find a rotation matrix and linear translation vector relating the two coordinate systems plus an addition Cartesian translation vector, defining the true location of the EM tracker in relation to the frame on the Gen2 kinematic chain where the sensor ideally would be placed. The cost function is thus a function of 15 parameters: 9 parameters to describe the rotation matrix \( R_{\text{optim}} \), 3 parameters for the translation vector relating the Aurora to the Gen2 coordinate system, \( t_{\text{optim}} \), and 3 parameters to describe the translation vector between the ideal and true EM sensor locations, \( e_{\text{EM}} \). The rotation matrix and translation vector from the previous method, \( M_{\text{Aurora}}^* \) and \( M_{\text{Aurora}}^* \), are used as the initial conditions for \( R_{\text{optim}} \) and \( t_{\text{optim}} \). The initial condition for \( e_{\text{EM}} \) is a zero vector. Should the EM tracker have been placed perfectly at the coordinate frame in the kinematic chain, the solution of the optimization problem would yield \( e_{\text{EM}} \) as a zero vector, \( R_{\text{optim}} = M_{\text{Aurora}}^* \) and \( t_{\text{optim}} = M_{\text{Aurora}}^* \). The cost function for the optimization problem is given in (3.14).
\[ J = \sum_{i=1}^{15} \left( (R_{\text{optim}} \cdot X^i_{\text{Aurora}} + t_{\text{optim}}) - (X^i_{\text{Aurora}} + e^0R_{\text{EM}} e_t) \right)^2 \]  \hspace{1cm} (3.14)

Constraints must be enforced so that the solution to the problem makes sense geometrically, i.e., the columns of the resultant rotation matrix are orthonormal. Let \( r_{i,j} \) represent the element in \( i \)-th row and \( j \)-th column of the rotation matrix returned by the optimization problem. The optimization problem with constraints is described by (3.15).

\[
\begin{align*}
\min_{x_{1,\cdots,15}} & \quad J \\
\text{s.t.} & \quad \lVert r_{1,1} + r_{2,1} + r_{3,1} \rVert = 1 \\
& \quad \lVert r_{1,2} + r_{2,2} + r_{3,2} \rVert = 1 \\
& \quad \lVert r_{1,3} + r_{2,3} + r_{3,3} \rVert = 1 \\
& \quad [r_{1,1} \quad r_{2,1} \quad r_{3,1}]^T \cdot [r_{1,2} \quad r_{2,2} \quad r_{3,2}]^T = 0 \\
& \quad [r_{1,1} \quad r_{2,1} \quad r_{3,1}]^T \cdot [r_{1,3} \quad r_{2,3} \quad r_{3,3}]^T = 0 \\
& \quad [r_{1,2} \quad r_{2,2} \quad r_{3,2}]^T \cdot [r_{1,3} \quad r_{2,3} \quad r_{3,3}]^T = 0
\end{align*}
\] \hspace{1cm} (3.15)

The constraints are nonlinear, thus requiring a nonlinear solver. Sequential quadratic programming was used by calling the \textit{fmincon} function from the MATLAB optimization toolbox [44].

### 3.1.5 Brachytherapy Needle

A 16-gauge brachytherapy needle was used to perform the experiments. This needle did not contain any LDR pellets. The needle was inserted into the gelatin phantom. An electromagnetic position sensor was embedded within the tip of a thin needle which was used to mark the target for each trial of the experiment. Figure 3.6 and Figure 3.7 illustrate the process of placing the EM trackers within a hollow needle.
3.1.6 System Architecture

The experimental setup can be seen in Figure 3.8. The entire system architecture for the equipment used in the experiments is shown in Figure 3.9. A Simulink model was created that featured all the custom library blocks to communicate with the hardware and all the appropriate signals were connected to create the control loops. The control algorithms were designed using Simulink. QUARC (from Quanser Consulting Inc.) [45] was used to generate real-time code from the Simulink model and was run with the highest priority on the host PC. Communication with the Gen2 and ATI F/T sensor operated at 100Hz, while communication with the Aurora NDI system operated at 40 Hz which was the hardware
limit. A custom Graphical User Interface (GUI) was designed using the Simulink Dashboard library to control the experimental setup. The GUI can be seen in Figure 3.10. The GUI featured options for the user to switch control modes, enable/disable data collection, perform an insertion, calibrate the Gen2 and Aurora coordinate systems, and an emergency stop. The GUI communicated with the real-time executable from QUARC via the UDP protocol at the computer’s local address.

![Figure 3.8: Experimental Testbed Setup](image)
Figure 3.9: Experimental Setup Architecture

Figure 3.10: Experimental Setup Control Graphical User Interface (GUI)
The top-level view of the Simulink block diagram can be seen spread across Figure 3.11, Figure 3.12, Figure 3.13 and Figure 3.14. The block diagram that controls the autonomous data collection routine (more details given in Chapter 4) is shown in Figure 3.15 and Figure 3.16.

![Simulink controller state machine](image)

**Figure 3.11: Simulink controller state machine**
Figure 3.12: Simulink needle controller inputs and outputs
Figure 3.13: Kinova Gen2 logic block diagram. This block diagram includes workspace boundary checking, target trajectory switching and cartesian to joint-space velocity mapping.
Figure 3.14: Simulink custom library blocks for hardware
Figure 3.15: Simulink block diagram for autonomous data collection (Part 1)
Figure 3.16: Simulink block diagram for autonomous data collection (Part 2)
3.2 Setup of the Experimental Procedure

This section outlines the steps required to set up and run each experiment. The specific manipulations of the needle vary from experiment to experiment, depending on which algorithm is being evaluated. The calibration procedure was performed while there were no objects resting on the surface of the EM field generator. Once the calibration procedure was complete, the gelatin phantom was placed on top of a plastic riser so that the phantom was in a position in the measurement volume and where the Gen2 could perform the insertions with a low likelihood of being in a kinematically singular configuration. Once the phantom was in place, the Gen2 was placed in gravity compensation mode so that the arm could be maneuvered to roughly the correct position, and orientation, required to begin the insertion. If an experiment was focused on steering the needle to a target, the user would press the “AlignNeedle” button in the GUI. This would cause the Gen2 to align the needle so that the axis along the needle coincides with the target location. Once aligned, the user was required to select the “EnterInsertionMode” button on the GUI. This button switches the Gen2 into joint velocity control mode and zeroes the force/torque measurements. After confirmation of a successful mode switch, the user would press the “PerformInsertion” button on the GUI, which passes control of the needle to the needle insertion algorithm. At this point the equipment runs autonomously, performing whichever task was specified by the user. The user can intervene by pressing the emergency stop button which will pause the robot in its current position and allow the user to either move the robot using the GUI or switch the manipulator back into gravity compensation mode. Each experiment is explained below.

3.3 Evaluation of Bezier Curve Kalman Filter

The aim of the first experiment was to evaluate how the proposed Bezier curve Kalman filter compares with a regular constant velocity KF. Needles were inserted into the gelatin phantom to a depth between 15 cm and 20 cm. The NDI system was used to record position measurements of the EM sensor located in the tip of the needle. Post processing was done to fit a curve to the sequence of needle tip position measurements after the insertion was complete. This curve was used to approximate the true trajectory of the needle tip. The approximated trajectory was then used as a reference to evaluate the performance of the
BC KF and regular constant velocity KF. Both filters had to be tuned. The constant velocity KF was tuned, finding that a measurement noise variance of 0.001 and a velocity process noise variance of 0.0001 resulted in the best observed performance. The process noise matrix was found using the same method as in (2.26) and (2.27). The optimal parameters for position and velocity measurement variances in the BC KF were found to be 0.01 and 0.0001 respectively. The process noise variance for position and velocity were both determined to be 0.0001. After the first insertion trial on the experimental setup, it was found that the velocity from the BC tangent was quite poor until the needle was at a depth of around 7 cm. This can be attributed to the fact that the tangent of the curve is very susceptible to noise during the early phase of the insertion. A low-pass filter was used to smooth the tangent estimates. This introduced one additional parameter to tune, the cut-off frequency of the low-pass filter. It was found that having a cut-off frequency of 2 rad/s resulted in the best performance. An example of the raw, and low-pass filtered velocity estimates from the BC tangent can be seen in Figure 3.17. There were 15 needle insertions performed to evaluate the proposed filter. The evaluation metric used to evaluate both filters was the running RMSE of the filtered position estimate. It was observed that the BC KF showed a 55% decrease in running RMSE when compared to the regular constant velocity KF. The standard deviation of the BC KF running RMSE was however over 2.5 times greater than the regular KF. This is likely because the BC KF is more prone to inaccurate velocity measurements in the early stages of the insertion, making for a greater variety in performance. Comparison of performance is plotted in Figure 3.18.
3.4 Trajectory Prediction Evaluation

3.4.1 Evaluation of Bezier Curve Integration Prediction Method

The purpose of the experiments outlined in this section was to evaluate how accurately the BC derivative integration method can predict the future trajectory of a needle tip, while also determining any practical limitations. The metric used for evaluation was the
Euclidean norm between the end of the predicted trajectory and the true final needle-tip position.

Early trials showed that despite the efforts made to make the curve fitting robust to noisy measurements, the predicted trajectory was often not feasible, sometimes turning with a radius much sharper than a needle can make, eventually turning a full 180 degrees. An example of a poor-quality trajectory prediction can be seen in . The blue curve shows the predicted trajectory if the needle were to continue being inserted with no rotation. The red curve shows the predicted trajectory if a 180-degree rotation were to take place at the current location of the needle tip.

![Figure 3.19: Poor quality trajectory prediction](image)

To overcome these inaccurate trajectory estimates, it was decided to instead use the filtered position measurements from the output of the BC KF to populate the position buffer, as opposed to the raw position measurements from the Aurora NDI system. This change greatly improved performance and reduced the occurrence of non-feasible trajectory predictions.

A set of 60 insertions were performed to evaluate how different combinations of prediction depth, rotation angle, and target depth affect the accuracy of the predicted trajectory. Prediction depth refers to the depth of the needle tip when the future trajectory is predicted;
target depth is the final depth of the needle tip; and rotation angle is the angle which the Gen2 rotates the base of the needle. The ranges of parameters used in these initial trials for prediction depth, target depth and rotation angle were [60, 120] mm, [150, 200] mm, and [0, 180] degrees respectively. It was found that the accuracy of the trajectory prediction rapidly declines when the prediction is made prior to the needle tip being at a depth of less than 10 cm. Unsurprisingly, it was shown that the accuracy increases as the gap between the prediction depth and target depth decreases, i.e., the prediction horizon shortens. The prediction error of the final needle-tip position is plotted against prediction depth and target depth in Figure 3.20. The reason for this sudden drop in accuracy is because the curve is very susceptible to overfitting noise in the position measurements, despite the efforts made to create an optimization problem that is robust to overfitting.

![Figure 3.20: Needle-Tip Error vs Target and Prediction Depth](image)

The position error is plotted against two other pairs of independent variables, namely, rotation angle and target depth and rotation angle and prediction deep in Figure 3.21 and Figure 3.22 respectively. The gradient of the prediction error with respect to the rotation angle was relatively small when compared to the gradient with respect to the target/prediction depths. This observation suggests that prediction depth and target depth are better metrics for estimating the expected trajectory prediction accuracy.
Figure 3.21: Needle-Tip Position Error vs Prediction Depth and Rotation Angle

Figure 3.22: Needle-Tip Position Error vs Target Depth and Rotation Angle

Acknowledging the limitation of the prediction algorithm, further trials were performed to evaluate the trajectory prediction accuracy, in a scenario where the needle was at least 10 cm deep prior to performing the trajectory prediction. The needle manipulation for each insertion adhered to the following sequence. The needle was inserted linearly to a depth in the range [100,120] cm before pausing linear insertion to rotate by an angle in the range [0, 180] degrees, then resuming linear insertion to the target depth. The predicted trajectory
was calculated using the known amount of rotation. An additional 60 insertions were performed. The RMSE of predicted needle tip location across all 60 trials was 1.44 mm with a standard deviation of 0.50 mm. The RMSE of the predicted needle tip positions can be seen plotted as a surface against prediction depth and target depth in Figure 3.23.

![Graph showing needle-tip position error vs prediction and target depth.](image)

**Figure 3.23: Needle-Tip Position Error vs Prediction and Target Depth**

Figure 3.24 and Figure 3.25 show the prediction error plotted against the rotation angle and the prediction depth, and the rotation angle and the target depth, respectively. An interesting observation is that the most accurate predictions were made when the needle rotated 180 degrees, while the least accurate rotations were when the needle rotated approximately 90 degrees. The prediction depth in the range [100 120] mm was shown to have little influence on prediction accuracy, while the prediction error grew roughly linearly with an increase in the target depth.
Figure 3.24: Needle-Tip Position Error vs Rotation Angle and Prediction Depth

Figure 3.25: Needle-Tip Position Error vs Rotation Angle and Target Depth

Figure 3.26 shows the predicted trajectory with a dashed blue line, for an insertion where the prediction was made at a depth of 120 mm, and the needle was inserted to a depth of 180 mm. Figure 3.27 illustrates the predicted trajectory in the scenario where the needle is inserted to a depth of 120 mm, rotated by 180 degrees, before the linear insertion continued to a depth of 180 mm.
3.4.2 Evaluation of Savitzky-Golay Trajectory Prediction Method

The experiments outlined in this section serve the same motivation as those in the previous section. On a gelatin phantom, 60 insertions were performed with prediction depths and rotation angles in the range \([40, 120]\) mm and \([0, 180]\) degrees respectively. As with the
Bezier Curve method, the SG-filter method demonstrated a sharp drop-off in trajectory prediction accuracy once the prediction depths were less than a certain amount, in this case 80 mm. This drop-off in accuracy can be seen by the final trajectory position error plotted as a surface against prediction depth and target depth in Figure 3.28.

Figure 3.28: Needle-tip position error using the SG-Filter over a large range of target and prediction depths

Further analysis was done, restricting the data to insertions that had a prediction depth greater than or equal to 80 mm. The final trajectory position error plotted as a surface against prediction depth and target depth, for the restricted domain can be seen in Figure 3.29.
Figure 3.29: Needle-tip prediction error surface using the SG-Filter on a restricted domain of prediction depths

Similarly, the trajectory prediction error is plotted against two other combinations of independent variables, namely rotation angle and prediction depth as well as rotation angle/target depth - see Figure 3.30 and Figure 3.31.

Figure 3.30: Needle-tip prediction error using the SG-Filter on restricted prediction depth domain. The surface is plotted against needle rotation and prediction depth
In these results, the relationship between accuracy and rotation angle is worth noting. When the prediction depth is closest to 80 mm, the SG-filter method is most accurate at predicting trajectories where a 180-degree rotation is performed, whereas when the prediction is made closest to 120 mm, the prediction is most accurate for trajectories with no rotation. The RMSE of the needle tip predicted position for the SG-filter is 3.43 mm with an SD of 0.76 mm. The SG-Filter method is the least accurate of the two proposed methods, however, it was able to make predictions consistently at smaller depths when compared to the BC method.

3.5 Closed-Loop Control Results

Experiments were performed to evaluate how well the proposed algorithms were able to steer a needle to a desired target. To recap, this requires an optimizer to continuously search for the best available trajectory that can be reached from the current position of the needle tip. Optimality is defined by minimizing the Euclidean distance between the predicted final location of the needle and the target location. Separate experiments were run using the two different trajectory prediction methods inside the optimization cost function. The target position was specified using a separate needle that too had an EM tracker located inside
the tip. The target location was set to be between 15 and 20 mm for each trial. Once the target location was set, the Gen2 was moved into place and oriented so that the axis of the needle coincided with the target location. Once in the correct position/orientation, the operator would press the insert needle button and allow the autonomous algorithm to assume control of the needle manipulation.

### 3.5.1 Bezier Curve Closed-Loop Performance

Sixty insertions were performed in a closed-loop fashion with the Bezier curve trajectory prediction method being used to search for optimal trajectories online. LDR brachytherapy is most effective when the accuracy of the final seed placement is within 5 millimeters [46]. The minimum predicted accuracy that was required for the high-level logic to act on an optimal trajectory candidate was set to be 3 mm such that the target had a margin of error with respect to the required accuracy. Figure 3.32 shows the distribution of the final position error using a histogram.

![Histogram showing distribution of final position errors](image)

**Figure 3.32:** Histogram showing distribution of final position errors when the acceptable threshold is set to 3 mm.

For further analysis, the final position errors were plotted against their respective rotation angles. This is shown in Figure 3.33. The red X’s denote attempts that failed to reach a
final position within the acceptable threshold, while the blue O’s denote attempts that were successfully steered to within the acceptable threshold.

![Scatter plot of the final position errors plotted against the optimal angle determined by the optimizer. The data points of zero rotation are instances where no solution was found.](image)

**Figure 3.33:** Scatter plot of the final position errors plotted against the optimal angle determined by the optimizer. The data points of zero rotation are instances where no solution was found.

There are a couple observations to note from these plots. The first is that the highest errors occurred when zero rotation was performed. Hence the solver never found an optimal trajectory candidate during the needle intervention, thus the Gen2 did not perform a manipulation to correct the trajectory. Another observation is that most of the trials resulted in the optimizer finding an optimal amount of rotation to be in the neighborhood of ±180 degrees. This is as expected as the needle begins to orient such that it points directly at the target location and the asymmetric needle tip begins steering the needle off the direct path, thus requiring a 180-degree rotation to steer the needle back to the target.

### 3.5.2 Savitzky-Golay Closed-Loop Performance

Sixty insertions were performed using the Savitzky-Golay closed-loop strategy. The threshold for the high-level logic was set to 3 mm as with the Bezier curve method. The SG approach was less accurate when compared to the BC approach, having a mean final
position error of 5.2 mm with a standard deviation of 2 mm. A histogram can be seen in Figure 3.34, illustrating the distribution of the final position accuracy from the trials.

![Figure 3.34: Histogram showing the distribution of the final position errors when the acceptable threshold is set to 3 mm.](image)

As with the BC approach, the majority of the optimal trajectory candidates required needle manipulation in the neighborhood of ±180 degrees, as shown in Figure 3.35.
Figure 3.35: Scatter plot of the final position errors plotted against the optimal angle determined by the optimizer.

Figure 3.36 shows the velocity and acceleration estimates found using the adaptive LSTM-SG filter during an insertion where no rotation was performed.

Figure 3.36: Velocity and acceleration estimates using the adaptive LSTM-SG filter.
While the velocity trended upwards during the insertion, the acceleration had trouble converging to a steady value. Because the second derivative did not converge to a steady-state value, the optimizer selected trajectory candidates that did not show the expected behavior. The trouble in converging to steady-state values can likely be attributed to errors in transforming the deflected needle-tip position into the end-effector frame of reference. Figure 3.37 shows the raw position measurements of the deflected needle-tip, that have been transformed into the end-effector frame.

Figure 3.37: Demonstration of the misalignment transforming the deflected needle-tip measurements into the end-effector frame.

A particular region of interest for this plot is the region where the linear movement (x-axis) is less than 2 cm. The needle was offset from the gelatin phantom by approximately 2 cm for each of the trials plotted. Despite the needle moving in free space for the first two cm of travel, the position measurements fluctuated on the order of 5 mm in some cases before ever contacting the gelatin phantom. There are several sources of error for this non-perfect transformation including nonlinearities in the Aurora NDI system, considering the Gen2 joints to be perfectly stiff, and assuming the kinematic description of the Gen2 and the custom 3D printed end-effector to be exact.
Chapter 4

4 Steering of Flexible Needles using an LSTM Encoder with Model Predictive Control

4.1 Machine Learning Review

A subset of neural networks is a class of networks called recurrent neural networks (RNNs). RNNs have memory, allowing them to detect temporal patterns in data sequences which they are trained on. A diagram of a vanilla RNN cell can be seen in Figure 4.1.

![Vanilla RNN cell](image)

**Figure 4.1: Vanilla RNN cell**

The basic RNN cell concatenates the output from the previous time step with the input to the cell from the current time step. The main drawback with the vanilla RNN cell is that it suffers from the vanishing gradient problem. Recall that RNNs are used to capture temporal information and thus receive sequences of input data. With RNNs, not only does the gradient need to propagate back through all the layers but also needs to propagate back through each timestep in the sequence. From the point of view of the BP algorithm, this effectively increases the number of layers which the gradient needs to propagate through by a factor equal to the number of timesteps in the sequence. The increase in layers that BP is required to propagate through can be visualized by unrolling the network (see Figure...
4.2). The vanishing gradient in RNNs causes time steps early in a sequence to have very little influence on the trainable parameters. The vanishing gradient problem is attenuated by using a variation of RNNs called Long Short-Term Memory (LSTM) networks.

![Unrolled RNN](image)

**Figure 4.2: Unrolled RNN**

LSTM networks are neural networks that contain what are known as LSTM cells. An LSTM cell is shown in Figure 4.3. The long-term memory in an LSTM cell is encoded into a vector which can be seen labelled as the cell state in Figure 4.3. The short-term memory is a vector labelled as the hidden state in Figure 4.3. Each of the gates in Figure 4.3 is a smaller neural network made up of regular neurons.

![An LSTM cell](image)

**Figure 4.3: An LSTM cell**
Within each LSTM cell there are thus 4 smaller neural networks. For the input at each timestep, the forget and input gates update the cell-state to reflect the input data. The forget gate determines the changes that should be made to the cell state such that data which is no longer important is forgotten. The input gate determines what information should be added to the long-term memory/cell state. Lastly, the output gate is responsible for extracting information from the cell state that is most relevant to the LSTM cell output for the current timestep. The output of the LSTM cell is the output from the output gate and is thus what is passed to other LSTM cells in the network. These LSTM cells can be stacked to form layers containing multiple cells, while layers comprised of LSTM cells can be connected sequentially to form deep LSTM networks in the same way that layers of regular neurons can be connected to form deep networks. The cell-state is the feature of LSTM cells that solves the vanishing gradient problem that vanilla RNN cells face. The gradient propagates along both the lower path, hidden state, and the upper path, cell state, thus there are two contributions to the gradients of all the trainable parameters in an LSTM cell. The cell state path provides a route for the gradient to propagate, layer-to-layer, timestep-to-timestep, without passing through an activation function. This pathway provides the gradients a route to travel that avoids exponential decay as it propagates through layers and timesteps, thus enabling data from the beginning of a training sequence to make a meaningful contribution to the values of the trainable parameters.

4.2 Introduction

Subcutaneous needle insertion is a minimally invasive procedure that is used commonly in procedures for biopsies and cancer treatment methods such as brachytherapy [17], [47]. The success of these procedures depends on the accuracy that is achieved, i.e., the proximity of the final needle tip position to the intended target. The major limitation of these procedures comes from the difficulty created by the deflection that the needle experiences when inserted into soft tissue [6], [48]. Bevel-tip needles are commonly used as the asymmetric tip offers a predictable direction of deflection [49]. This predictable direction of deflection allows surgeons to employ different rotation strategies to correct for the deflection and steer the needle to the intended target. One of the most common strategies is to perform a series of 180-degree rotations over the course of the insertion.
The alternating direction of the bevel tip reduces the net deflection from the ideal path over the course of the insertion [15]. One disadvantage of this rotation strategy is that the accumulation of rotations causes tissue damage and patient discomfort [50]. More complicated control strategies require very precise manipulations that are difficult and/or impossible to be carried out by hand. Several research groups have designed novel manipulators for needle insertion applications, e.g., [6], [24]. The authors of [24] designed and built a custom 5 degrees-of-freedom (DOF) manipulator specifically for use in percutaneous needle insertion. The manipulator made the control problem easier as the assembly featured back drivable joints, stationary actuators, and redundant sensors, all of which contribute to enhanced safety. Many groups have suggested control strategies, for needles in subcutaneous procedures, that can be performed with the mentioned robotic systems. Some of the earliest work, presented in [25], used unicycle and bicycle kinematic models to describe the motion of a bevel-tip needle in soft-tissue. These kinematic models consider the needle to be driven by two velocity inputs, linear and rotational. This work however does not consider interaction forces between the needle and tissue. The needle and soft tissue behave as a coupled system during an insertion procedure, so considering only the kinematics is insufficient to fully describe the interaction. A complete dynamic description of the system requires an interaction force measurement between the needle tip and soft tissue. While obtaining such measurements presents a variety of challenges, it was shown in [29] that forces measured at the base of the needle can be used to estimate the load acting on the inserted needle tip. In [51], optical sensors were used to measure the cutting force. Interaction force measurements enable the use of mechanics-based models to predict needle deflection. In [29], Euler-Bernoulli beam theory was used to develop an adaptive quasi-static model of a needle during deflection that can accurately predict needle deflection and shape. In [52], mechanics-based modelling was used to estimate the shape of a needle that has undergone multiple rotations during the insertion process. However, this work was limited to planar motions. Additionally, this method used a series of 180-degree rotations to compensate for the deflection of the needle. As noted in [50], the accumulation of needle rotations during a procedure can result in tissue damage. One drawback with the mechanics-based models is that prior assumptions must be made about the system. One common assumption is that the only source of deformation is the needle
and that the tissue itself does not deform. While the deflection and shape of the needle are important metrics for the procedure, the work presented in [29] and [52] only estimates the deflection and shape for the current state of the needle and does not offer estimates of future deflection and shape. Predicting the future state of the needle, in response to current and future needle manipulations is important for accurately steering the needle to the desired target. Work has been done in [32] and [53] to create dynamic models of needle-tissue interaction that capture properties such as friction. These dynamic models can be used to predict the future trajectory of needles for varying control input sequences. Developing a generalizable model for needle-tissue interaction proves difficult due to the non-homogeneous property of tissue and the fact that the needle will pass through several different types of tissue during a typical procedure. The increased interest in machine learning (ML) over the past decade has provided an opportunity to use modern ML tools to develop improved needle-tissue interaction models. The flexibility of modern ML tools, such as neural networks, can facilitate the modeling, control, and trajectory planning of needle insertion by learning needle-tissue interaction dynamics from collected data.

This chapter presents a method for using recurrent neural networks (RNNs) to predict an optimal set of needle manipulations, requiring only a single rotation to steer a bevel-tip needle to the desired target. The first contribution of this work demonstrates the ability of RNNs to estimate the future needle trajectory using reaction forces measured at the needle base, and position measurements of the needle tip. This method makes no assumptions about needle-tissue interaction. Instead, it uses real data to find a relationship between measurements and future trajectories. The second contribution of this work demonstrates how a transformation on the latent measurement space enables a neural network to predict the future trajectory of the needle in response to current and future manipulations. This contribution allows an RNN in combination with a model predictive controller (MPC) to predict an optimal trajectory, and the needle manipulation required to minimize the number of rotations taken over the duration of the needle-based intervention while accurately steering the needle to the desired target.
4.3 Methodology

4.3.1 Collecting Experimental Data for Modeling

Needle insertions were performed on a gelatin phantom tissue. The experimental setup consisted of an 16-gauge brachytherapy needle attached to a custom 3D printed end-effector (EE), affixed to a 6 degrees-of-freedom (DOFs) manipulator. A force/torque sensor was integrated within the 3D-printed end-effector to measure the forces and torques at the base of the needle. An electromagnetic (EM) position sensor was embedded in the needle tip to measure the deflection. The phantom tissue was placed on a tabletop EM field generator. The coordinate systems of the robotic manipulator and the EM field generator were aligned. The transformation relating the base frame to the EE was used to compute the needle tip position with respect to the EE frame. The insertions were performed on the phantom with a constant linear velocity of 2.5 cm/s.

4.3.2 Needle Insertion Model

Functions mapping the insertion depth to \( x \) and \( y \) coordinates of the needle-tip deflection (EE frame) were obtained to create a kinematic model for needle insertion. Similarly, functions mapping \( x \) and \( y \) deflections (EE frame) to the force/torque measurements were used to create a mechanics model of needle-tissue interaction. The functions were obtained by finding the best fitting functions for the data from several different insertions. This was done using the MATLAB Curve Fitting Toolbox. A list of the curves used is given in Table 4.1.
<table>
<thead>
<tr>
<th>Output</th>
<th>Domain</th>
<th>Fitted Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-Axis Deflection</td>
<td>Insertion Depth</td>
<td>Rational Function Numerator/Denominator degree =5/4</td>
</tr>
<tr>
<td>Y-Axis Deflection</td>
<td>Insertion Depth</td>
<td>Rational Function Numerator/Denominator degree =5/4</td>
</tr>
<tr>
<td>X-Axis Force</td>
<td>X-Axis Deflection</td>
<td>Rational Function Numerator/Denominator degree =5/5</td>
</tr>
<tr>
<td>Y-Axis Force</td>
<td>Y-Axis Deflection</td>
<td>Rational Function Numerator/Denominator degree =3/4</td>
</tr>
<tr>
<td>Z-Axis Force</td>
<td>Insertion Depth</td>
<td>First Degree Polynomial</td>
</tr>
<tr>
<td>X-Axis Torque</td>
<td>X-Axis Deflection</td>
<td>Rational Function Numerator/Denominator degree =4/2</td>
</tr>
<tr>
<td>Y-Axis Torque</td>
<td>Y-Axis Deflection</td>
<td>Rational Function Numerator/Denominator degree =4/4</td>
</tr>
</tbody>
</table>

**Table 4.1: Deflection Model Curve Fitting**

A kinematic model was created to describe the trajectory of the needle with respect to a base frame. The kinematic model of the needle motion has two inputs: linear velocity \( v_{\text{Tip}} \) and rotational velocity \( \omega_{\text{Tip}} \). These control inputs are applied to the base of the needle, along and about the z-axis respectively, during insertion and directly control the simulated trajectory. The reaction forces, acting at the base of the needle are calculated using the functions from Table 4.1. Figure 4.4 depicts the arbitrary base frame and the frame attached to the moving needle tip.

![Figure 4.4: Frames for the Needle Kinematic Model](image)
4.3.3 Dataset Generation

The needle insertions were simulated according to the strategy shown in Algorithm 4.1.

**Begin Simulation**

**Initialization**

Generate a random time, $T_1$, in the range $[1.5, 3.5]$ seconds.

Generate a random rotation amount, $\Theta$, in the range $[-180, 180]$ degrees.

**While** (Time$<$T1)

   Insert needle linearly at a velocity of 2.5 cm/s

**End**

**Pause Linear Motion**

**Rotate $\Theta$ Radians**

**While** (Insertion depth$<$Max Insertion Depth)

   Insert needle linearly at a velocity of 2.5 cm/s

**End**

**End of Simulation**

**Algorithm 4.1: Description of control input sequence during data collection**

The needle tip positions in the base frame and in the EE frame, the force/torque, and manipulation type were all recorded at 10Hz. The manipulation type is a piecewise function described by (4.1). The random rotation amount ‘$\Theta$’ for each simulation was also recorded. The deflection measurements in the EE frame are denoted by $^{ee}x$ and $^{ee}y$. One additional simulation was performed without any rotation: the needle was inserted without any rotation until it reached the maximum insertion depth. The position data from this additional simulation is later used as the network target for time steps prior to the rotation occurring. The data from this additional simulation is needed because the network should not predict a change in the trajectory prior to the rotation occurring. Let the deflection of the needle tip when no rotation occurs, with respect to the EE frame, be denoted by $^{ee}\bar{x}$ and $^{ee}\bar{y}$.

$$f(x) = \begin{cases} 
-1, & \text{rotating negative direction} \\
0, & \text{linear insertion} \\
1, & \text{rotating positive direction}
\end{cases} \quad (4.1)$$
4.3.4 Trajectory Prediction Network

All measurements were normalized to have a zero mean and standard deviation of 1. The data was then split into groups: training (80%), validation (15%) and testing (5%). Some nomenclature of the prediction horizon must be given before continuing. The prediction horizon is a discrete number of points along the predicted trajectory. Let \( N \) denote the number of points in the prediction horizon and \( M \) denote the number of time steps between successive horizon points. For example, for measurements captured at 10Hz, with \( M=3 \), each point in the prediction horizon is separated by 0.3s.

The measurements used as network inputs for training are shown in (4.2).

\[
Input_t = \begin{bmatrix} ee_x \\
               ee_y \\
xForce \\
yForce \\
zForce \\
xTorque \\
yTorque \\
zTorque \\
ManipulationType \end{bmatrix}
\]  \hspace{1cm} (4.2)

Concatenating the measurements from multiple timesteps resulted in better performance. It was determined that using measurements from nonadjacent timesteps further improved performance. This is likely because the dynamics of the needle-tissue interaction happen at a slower rate than the measurement sampling rate. The optimal number of time steps between each set of measurements in the buffer was determined to be 5. The final input vector to the network is then given in (4.3).

\[
StackedInput = \begin{bmatrix} Input_t \\
                               Input_{t-1+M} \\
                               \vdots \\
                               Input_{t-(N-1)+M} \\
                               Input_{t-N+M} \end{bmatrix}
\]  \hspace{1cm} (4.3)

The delayed inputs were zero-padded to fill the timesteps at the beginning of the sequence where measurements were not yet available. Let \( L \) be the total length of the simulation.
The network input sequence was the stacked input vector for time steps in the range \([1, L-N*M]\). This limit is in place because if the input domain exceeds \(L-N*M\) time steps, there would not be targets available for training as the simulation would not have run that long.

The network was trained to predict the \(x\) and \(y\) deflections of the needle tip with respect to the EE frame. The training targets for time step \(t\) are calculated according to Algorithm 4.2. This algorithm determines whether the prediction horizon should be populated with points from a future trajectory that has undergone rotation, or whether it should be populated with target points that assume the needle continues with linear insertion. For timesteps prior to the timestep where rotation begins, the network has no indication that rotation is going to take place so it should make prediction only on what it has observed and thus make a prediction assuming that the needle is continuing with linear manipulation for the remainder of the insertion. Once rotation has taken place, then the needle should predict a trajectory in response to that rotation. Note that by examining the sequence of manipulation type measurements, the index of time steps is known prior to, and after rotation. Let these two time steps be denoted by \(n1\) and \(n2\).

\[
\text{If } (t<n1) \quad \text{target} = \begin{bmatrix}
 ee_x t+M+1 \\
 ee_x t+M+2 \\
 \vdots \\
 ee_x t+M*(N-1) \\
 ee_y t+M+N \\
 ee_y t+M+1 \\
 ee_y t+M+2 \\
 \vdots \\
 ee_y t+M*(N-1) \\
 ee_y t+M+N
\end{bmatrix}
\]

\[
\text{If } (t>n1) \quad \text{target} = \begin{bmatrix}
 ee_x t+M+1 \\
 ee_x t+M+2 \\
 \vdots \\
 ee_x t+M*(N-1) \\
 ee_x t+M+N \\
 ee_y t+M+1 \\
 ee_y t+M+2 \\
 \vdots \\
 ee_y t+M*(N-1) \\
 ee_y t+M+N
\end{bmatrix}
\]

Algorithm 4.2: Decision for which target prediction horizon to use, given the current timestep.
The trajectory prediction network in the proposed method is motivated by the LSTM-Encoder-MLP network architecture in [34]. The architecture of our network can be seen in Figure 4.5.

![Figure 4.5: The Trajectory Prediction Network](image)

The input vector is passed to an LSTM encoder. The encoder consists of three LSTM cells in series. Each cell has an output size one half of the size of its input. The first two LSTM cells pass their hidden state to the subsequent cell, the third LSTM cell outputs both the cell state and the hidden state. The current control input type is concatenated, via skip connection, with the output of the third LSTM cell. It was found that having this skip connection improved performance. The output of the concatenation layer is passed to a fully connected layer. The network has a second skip connection that connects the current position to the output of the fully connected layer. The summation layer following the fully connected layer adds the current $x$ and $y$ deflection to the appropriate outputs of the fully connected layer, i.e., the current $x$ deflection is added to the outputs indexed by the range $[1, N]$, $y$ deflection is added to the output indexed by the range $[N+1, 2*N]$. The skip connections do not have “learnable” weights; instead, they have constant weights of 1. The appropriate data is extracted from the input via the skip connection by modelling the skip connections as fully connected layers with zeros biases and sparse connection weight matrices. The non-zero elements of the connection weight matrices have values of one and are appropriately placed so that only the information of interest is passed along from the dense input vector. The ‘current position’ skip connection having non-learnable gains
forces the LSTM Encoder and fully connected layers to predict the relative locations of future trajectory points. This was done to avoid overfitting and to make the network more generalizable [34].

Training was performed on a single GPU via the MATLAB Deep Learning Toolbox. In order to ensure that the training provided satisfactory results, concatenating multiple timesteps was required. Using measurements from non-consecutive timesteps showed further improvement. It required several attempts to determine appropriate hyperparameters of the network. Once determined, the network parameters consistently converged to values which resulted in acceptable performance. The Adam optimizer was used with an initial learn rate of 0.001. L2-Regularization, gradient decay factor and squared gradient decay factor were used with values of 0.0001, 0.9 and 0.99 respectively. A standard RMSE loss function was used for backpropagation. Training was run for 200 epochs at which point the RMSE was 0.46. The RMSE of the validation data matched that of the training data.

4.3.5 Trajectory Prediction with Rotation Network

An additional feed-forward neural network was trained to perform a transformation on the latent space of the LSTM encoder. This network is referred to as the latent space transformation network (LSTN). The purpose of this network is to transform the latent space from time step $n_1$, to the latent space at time step $n_2$. The LSTN has two inputs, the encoder latent space, and $\beta$, which is the amount of rotation occurring between time steps $n_1$ and $n_2$. The inputs for $\beta$ during training are the random amounts of the rotation ‘$\Theta$’ described in Algorithm 4.1. The LSTN is important because, with enough training data, the LSTN enables the trajectory prediction network to explore all possible future trajectories that can be achieved by rotating the needle at its current position. The network architecture can be seen in Figure 4.6. The network has 3 fully connected layers in series. The rotation angle is concatenated with the latent space prior to being passed to the first fully connected layer. There is a skip forward layer that concatenates the rotation angle with the output of the first fully connected layer as this demonstrated improved performance, much like the case with the trajectory prediction network.
The training inputs and targets for the transformation network are computed using the LSTM Encoder layers from the trajectory prediction network. To generate the network input for a given sequence, measurements from time steps 1 through $n_1$ are passed through the LSTM Encoder. The known rotation angle is concatenated with the latent space of the LSTM Encoder from time step $n_1$. Likewise, measurements from time steps 1 through $n_2$ are fed through the LSTM encoder to generate the latent space for the LSTN network target. This process is done for all 1000 simulation sequences. The LSTN was trained on a single GPU using the MATLAB Deep Learning toolbox. The Adam optimizer was used with the same training options as described in the training for the trajectory prediction network. The network was trained for 12000 epochs. The standard RMSE loss function was used. The final RMSE loss for both the validation and training data was 0.11.

Predicting the effect on future trajectories due to needle rotation can be performed by combining the trajectory prediction network with the LSTN. The combined architecture can be seen in Figure 4.7. This network has two inputs, the time series measurements from (4.3) and $\beta$. By exploring different values of $\beta$, the network can predict the effect of rotation.
on the future trajectory. The optimizer in section 4.3.7 looks to find an optimal value of $\beta$ to determine an optimal control strategy.

4.3.6 Extrapolating Prediction Horizon

One issue that arises when working with a finite prediction horizon is that the prediction horizon may not venture far enough into the future to offer any meaningful prediction as to how close the trajectory will come to reaching the desired target. To address this issue, a curve is fitted to the predicted trajectory points so that the predicted trajectory can be extrapolated past the distal end of the prediction horizon. It was shown in [38] that a quadratic Bezier curve (BC) could accurately model continuum objects. One limitation of the work was that quadratic curves restricted the use to planar motion. To address this shortcoming of quadratic BC, a quartic BC was used for our application. Using the quartic BC enables the curve to accurately fit the predicted trajectory in the presence of non-planar motion. A quartic BC is a parametric curve denoted by $B(t)$. The parameter, $t$, lies in the range $[0,1]$. The curve begins at the point $P_0$ and ends at $P_4$. When $t=0$, $B(t)=P_0$, and when $t=1$, $B(t)=P_4$. There are 3 control points, $P_1$, $P_2$, and $P_3$. The equation of the curve is given by (4.4).

$$B(t) := (1 - t)^4 * P_0 + 4 * t * (1 - t)^3 * P_1 + 6 * t^2 * (1 - t)^2 * P_2 + 4 * t^3 * (1 - t) * P_3 + t^4 * P_4$$

Before fitting a BC to the predicted trajectory points, the points are transformed from the EE frame to the base frame. Recall that the Trajectory Prediction Network only predicts
the $x$ and $y$ deflection in the EE frame. As a result, the known linear insertion velocity is
used to generate future $z$ positions for each predicted horizon point with respect to the EE
frame. The EE frame predictions are package into a $3\times N$ array, $eePredictions$, shown in
(4.5). The points in (4.5) are transformed to the base frame coordinate system according to
(4.6).

$$eePredictions = \begin{bmatrix}
    eeX_{t+1} & \cdots & eeX_{t+N} \\
    eeY_{t+1} & \cdots & eeY_{t+N} \\
    eeZ_{t+1} & \cdots & eeZ_{t+N}
\end{bmatrix}$$ \hspace{1cm} (4.5)

$$\overset{0}{Prediction} = \overset{0}{e}R * eePredictions + \overset{0}{Current\ EE\ Position} \hspace{1cm} (4.6)$$

Fitting the curve to the predicted trajectory requires determining the control points of the
BC. Determining the location of the control points is done by minimizing a cost function.
The cost function for this optimization problem is defined as the sum of all norms between
the closest point along the BC to each prediction horizon point. The cost function is shown
in (4.7). Note that $t_n$ is the BC parameter for the $n$th prediction point and is also an
optimization parameter.

$$J = \sum_{n}^{N} ||H_n - B(t_n)||$$ \hspace{1cm} (4.7)

The high degree of a quartic BC can result in fitting the trajectory very well within the
parameter range $[0,1]$. However, if the predicted trajectory points do not fit along a smooth
curve, the BC can have unexpected behavior outside the normal parameter range of $[0,1]$.
Evaluating the curve when the parameter $t$ is greater than 1 is required to extend the BC
beyond the distal point in the prediction horizon. Therefore, constraints must be placed on
the solver to ensure realistic behavior outside the range $[0,1]$. The constraints are described
in (4.8) and (4.9) below, where $\Phi$ is the angle between the tangents of the BC evaluated at
$t=1$ and $t=1+D$, $D > 0$ and $\delta$ is the maximum allowable angle between the vectors described
previously.
\[ \phi^2 < \delta^2 \quad (4.8) \]

\[ \phi = \sin^{-1}\left[ \frac{\| \hat{B}(1 + D) \times \hat{B}(1) \|}{\| \hat{B}(1 + D) \| \| \hat{B}(1) \|} \right] \quad (4.9) \]

This inequality constraint ensures that the tangent of the curve beyond the normal range \([0,1]\) is not drastically different from the tangent at the end of the prediction horizon. This is an expected behavior of a needle passing through tissue. D and \(\delta\) are controller parameters that can be tuned to suit the needs for different needle insertion applications.

### 4.3.7 Model Predictive Controller

Model Predictive Control (MPC) is an optimal control strategy that relies on a system model to predict future states of the system over the duration of a prediction horizon. MPC determines a sequence of system inputs that minimize a cost function over the duration of the prediction horizon. The cost function typically weighs the importance of accurately tracking desired system states vs the cost of control inputs. The minimum of the cost function is found via numerical optimization methods that explore the value of the cost function for different sequences of control inputs. At each time step, MPC will determine a sequence of optimal control inputs that result in a prediction horizon with the minimum cost. The sequence of optimal control inputs is recalculated at each time step. The desired control inputs to the system at time step \(t\) is the first control input from the sequence of optimal control inputs calculated at time step \(t-1\).

The system model used by the MPC in our application is the trajectory prediction network combined with the LSTN. The cost function is defined as the norm between the closest point along the BC, fitted to the predicted horizon points, and the target location. The cost function only depends on the rotation angle input to the LSTN. This creates a simple optimization problem that can be easily solved using a numerical optimization algorithm. The MPC only executes the determined optimal rotation when the cost is below some specified threshold ‘C’, which is the upper bound on the acceptable distance between the target and the closest point on the predicted needle trajectory.
4.4 Simulation Results

Simulations were run to evaluate how accurately the network could predict future trajectories for two scenarios (a) when the needle was inserted linearly to the desired depth with no rotation, and (b), when the needle is rotated by some angle $\theta$ at a depth less than the final depth. The evaluation was then further extended to determine how well the network could be integrated within an optimization problem to determine an optimal control strategy which could steer the needle to the desired target. The Euclidean norm between the needle tip’s final prediction location and the true final location was used to quantify accuracy for the trajectory prediction evaluations. Similarly, the Euclidean norm between the target location and final needle tip location was used to evaluate the performance of using the NN within an optimization framework. The simulation is purely a kinematic simulation where the needle follows a non-linear path according to the behaviour outlined in section 4.3.2. It is acknowledged that Sim to real transfer is a very difficult problem because simulations often lack sufficient accuracy and stochastic nature for the trained networks to perform well in the physical world. With that said, the hopeful outcome was to demonstrate the NNs can be used to accurately predict how control actions taken at the needles current position affect the future state/trajectory of a system. Further, it is hoped that this demonstration shows the possibility of integrating NNs into control architectures for a broader class of systems.

Ten simulations were run for scenarios (a) and (b). The network accurately predicted trajectories in both scenarios. The RMSE between the needle-tip’s predicted location and true final location were 0.215 mm and 0.445 mm for scenarios (a) and (b) respectively. Figure 4.8 illustrates the prediction trajectory for both scenarios. The needle tip for both simulations begin at the origin frame. For scenario (a), the needle was inserted linearly to a depth of 200 mm. The green ‘O’ marks the spot where the prediction was made under the assumption that only linear insertion will occur. The trajectory of the needle prior to the trajectory prediction is shown by the dotted blue line. The prediction horizon, made when the needle tip was at the prediction location, is illustrated with the orange ‘Xs’. The solid dark blue line shows the needle trajectory post-prediction. For scenario (b), the needle was inserted linearly, as with (a), until it reached the prediction location marked by the green
‘O’. The needle was then rotated 90 degrees before continuing to a depth of 200 mm. For scenario (b), the prediction horizon is illustrated by the purple ‘Xs’ while the true path is marked by the solid yellow line.

Figure 4.8: Trajectory Prediction Demonstration

Ten simulations were run to evaluate the accuracy of the control framework that uses the trained network as the system model to determine an optimal moment in time to perform a needle rotation manipulation and by how much the needle should be rotated by. The constraint parameters used for the BC fitting in the MPC were $\delta=\pi/12$ and $D=2$. The cost-action threshold, $C$, was set to 0.0015, meaning that if a trajectory was predicted that would pass within 1.5 mm of the target, the controller would execute that trajectory. Figure 4.9(a)
illustrates the Bezier curve being used to extrapolate the prediction horizon when the prediction horizon does reach the target location.

![Diagram of Bezier curve and trajectory prediction](image)

**Figure 4.9: Closed Loop Performance with Trajectory Prediction**

(a) Bezier curve (blue) is used to extrapolate prediction horizon

(b) Final trajectory of needle.

The dotted blue line illustrates the path the needle has traveled. The green ‘O’ illustrates the location of the needle tip when the optimizer has found a potential optimal trajectory candidate. The prediction horizon is shown by the orange ‘Xs’, while the Bezier curve, which is fit to the prediction horizon and extrapolated, is shown by the solid dark blue line. The target location is shown by the green star. The optimizer expected that a rotation of 147 degrees will result in the needle being steered to within 1.19 mm of the target. Figure 4.9 (b) illustrates the final trajectory of the needle in the dotted blue line and again shows the target with the green star. The needle was steered to within 1.41 mm of the target. The RMSE of the 10 closed-loop simulations was 1.63 mm. Reasons why the closed-loop control did not perform as well as the simple trajectory prediction scenarios discussed
above can be attributed to the Bezier curve overfitting the prediction horizon and having active constraints to avoid erratic behavior for $t > 1$. The result of the constraints and overfitting is a Bezier curve that does not perfectly predict the future trajectory beyond the distal end of the prediction horizon.

### 4.5 Experimental Results

A dataset of over 1300 needle insertions was collected using the experimental setup described in Chapter 3. The purpose of this work was to evaluate the performance of the proposed LSTM AE on a real experimental setup and to evaluate how large of a dataset would be required to train such a network. Autonomous routines were developed for the Gen2 to perform data collection with limited reliance on a human operator. The operator would power-on the system and perform a calibration between the Gen2 and the Aurora NDI system. The Gen2 was then placed into gravity compensation mode so that the operator could manually move the end-effector, and subsequently the needle, into a position offset a few cm from the gelatin phantom surface such that the needle axis was roughly normal to the tissue surface. The manipulator was then switched into insertion mode. Prior to commencing each data collection run, the program would generate a grid of evenly spaced positions, centered at the current position of the end-effector frame and normal to the end-effector z-axis. The manipulator would move so that the end-effector frame coincided with the desired grid point before pausing prior to performing an insertion. During each insertion, the needle rotated by some angle $\theta \in [-\pi, \pi]$, at a depth $d \in [5,15]$ cm. Once the needle-tip had reached its target depth $d_{target} \in [15,20]$ cm, the needle briefly stopped, then retracted until the end-effector frame origin coincided with the grid point where the insertion had begun. The end-effector would reverse the rotation that was performed during the insertion to prevent cables becoming tangled. The manipulator would move through all grid points one-by-one until it had performed an insertion beginning at each point. Once the grid was complete, the manipulator would move back to the first point. Before continuing, the operator would slightly adjust the position of the gelatin phantom relative to the manipulator so that the needle could penetrate an un-touched patch of gelatin. This process was repeated until the gelatin had been cut enough times that the needle could no longer pass through an untouched volume of gelatin. Each gelatin phantom typically
could facilitate just over 100 insertions before it could no longer be used. Gelatin became noticeably softer as it warmed up, and as a result, the needle-tip was less susceptible to deflection and would often slide along a near straight line.

Similar measurements were recorded during the experimental data collection as when simulation data was gathered. A dataset was generated with a variety of different combinations of three parameters: (a) the depth when a rotation was performed: (b) the amount by which the needle rotated: and (c) the final depth to which the needle was inserted. The training dataset covered the full range of the mentioned parameters to ensure that while running on hardware, the network is interpolating between previously seen data points and does not need to extrapolate beyond the data used during training. The recorded measurements include needle tip position, with respect to the end-effector frame, and force/torque measurements at the base of the needle. Figure 4.10 illustrates an example of $x$ and $y$ position measurement sequences that were included in the training data. The $x$ and $y$ positions of the needle tip (shown in red and blue respectively) with respect to the end-effector frame are plotted against time. The solid line segments show the observed position measurements from all time steps up to and including, the current timestep. The X’s are upcoming positions sampled from later time steps. The data points that form the solid line segments are the input sequences for the LSTM, while the X’s are network output targets for that particular timestep. Similar to Figure 4.10, Figure 4.11 illustrates training data for the scenario when the needle has undergone rotation during the insertion. It should be noted how the $x$ and $y$ positions of the needle-tip with respect to the base frame exhibit sinusoidal motion when plotted against time over the time period $t \in [3.5,5]$ s. This domain in time is the period when the base of the needle is rotated by the robotic manipulator. When this happens, the position of the needle remains (approximately) constant in 3D space with respect to the ground frame; however the end-effector of the Gen2 is rotating so the relative position changes.
Figure 4.10: Example of position data from a sequence with no rotation used to train an LSTM network

Figure 4.11: Example of position data from a sequence with rotation used to train an LSTM network

The same network architecture that was described in section 4.3.5, was used for the experimental data. The data was split into a 70/25/5 train/validation/test split. Figure 4.12
and Figure 4.13 show the network predictions and true future positions plotted against time for instances when the predictions were poor and good respectively.

**Figure 4.12:** Poor performance of the network predicting future needle trajectory in the scenario where no rotation was performed

**Figure 4.13:** Good performance of the network predicting future needle trajectory in the scenario where no rotation was performed
For comparison, the same plot is shown in Figure 4.14, where predictions are made using simulation data and the network that was trained using the simulation data. The prediction horizon points in Figure 4.14 nearly perfectly coincide with the target points.

![Network Predictions and True Positions for no Rotation Scenario](image)

**Figure 4.14: Example of very good network prediction quality using simulation data**

By comparing Figure 4.12 and Figure 4.13 with Figure 4.14, it is apparent that the network does a much better job of predicting the future trajectory of the needle when using simulation data. There are many reasons why the network which used experimental data did not train as well as the network that used simulation data. The main reason is that the simulation data came from a purely kinematic model where interaction forces played no role in the trajectory of the needle. As a result, the path which the needle followed in simulation was a function of only the two velocity inputs (linear and rotational). The needle trajectory recorded using the experimental setup on the other hand was dependent on many inputs including, gelatin stiffness changing during the experiment due to temperature variation, gelatin deforming differently in different volumes due to previous needle insertions, Gen2 arm not moving in a perfectly straight line nor with a perfectly constant velocity, accumulated gelatin residue on the needle adding additional friction along the length of the needle, the misalignment issues mentioned in section 3.5.2, etc.
One possible solution to remedy the poorer quality of the experimental data network is to use a larger dataset to train the network. An objective of this research was to determine an estimate for how much data would be required for the experimental network to have comparable performance to that of the simulation network. There is significantly more variance in the experimental data so increasing the amount of training data would likely provide the network with a richer set of experiences to draw from when predicting future trajectories in a real experimental setup. Figure 4.15, Figure 4.16, and Figure 4.17 show the training progress using datasets from experimental data consisting of 400, 800 and 1300 needle insertions respectively.

Figure 4.15: Training progress of the network trained using experimental data from 400 insertions.

Figure 4.16: Training progress of the network trained using experimental data from 800 insertions.
Figure 4.17: Training progress of the network trained using experimental data from 1300 insertions.

The insertions used in the datasets of sizes 400 & 800 were randomly sampled from the entire dataset. There is a noticeable improvement when using larger data sets. The RMSE of the network trained with the entire dataset is 81% and 61% of the RMSE when trained using only 800 and 400 insertions respectively. The decreasing RMSE with respect to the size of the dataset suggests that continuing to collect more data would continue to lower the RMSE and improve the network performance. The training progress of the network trained using simulation data is shown in Figure 4.18. The RMSE during training for the simulation network repeatably converges to less than 30% of the RMSE from the experimental data network. For reference, the simulation dataset must be reduced by a factor of 50 to match the training progress of the experimental data. This observation is important because it shows that the same network was able to improve from having very poor performance to near perfect performance, by only increasing the amount of data used during training. This suggests that increasing the experimental dataset by a factor of order 50 may result in the experimental network improving to performance comparable with the simulation network. Related future work may include collecting a significantly larger dataset to evaluate how much the network performance may increase.
Figure 4.18: Training progress of the network trained with simulation data from 1000 insertions.
Chapter 5

5 Conclusion and Future Work

5.1 Summary

This thesis discussed a body of research focusing on methods used to predict needle trajectories in response to manipulations on a needle at its current location. A review of the literature pertaining to needle-based interventions was conducted in Chapter 1 to familiarize readers with the highly nonlinear interaction between a needle and surrounding tissue during needle-based interventions such as brachytherapy or tissue biopsies. Existing strategies and techniques to compensate for needle deflection were discussed, and limitations of the existing methods were highlighted. A gap in the research was identified to be a lack of adaptive methods that can predict needle trajectories. The lack of machine learning methods for control algorithms for needle-based interventions was also identified. This thesis therefore focused on exploring trajectory prediction methods while simultaneously looking for ways to integrate modern machine learning techniques into the research to address the inherent nonlinearities and uncertainties in the problem.

Chapter 2 presented state estimation and control algorithms that were designed such that all reachable needle trajectories could be explored in real-time during a needle-based intervention. The trajectory prediction methods were both designed in such a way that a numerical solver could be used to determine an optimal amount of needle rotation at each time step. Trajectories in response to needle manipulations were predicted using two new methods. The first method would fit a Bezier curve to the observed trajectory, then extrapolate the curve to predict the future trajectory. The second method transformed the deflected needle tip position into a coordinate system rigidly attached to the base of the needle. The first and second derivatives, with respect to the needle-base frame, were estimated using a new adaptive filter, namely the adaptive LSTM Savitzky-Golay filter. Obtaining accurate derivative estimates with respect to the needle-base frame was much more difficult than anticipated. This difficulty is what led to exploring the use of machine learning to improve state estimation for a conventional SG-filter. It is expected that the LSTM-SG filter could be used in other domains provided sufficient training data is
available. Both prediction methods explored the hypothesis that the second derivative of the needle tip motion, with respect to the needle base frame, was directly related to the orientation of the asymmetric tip of a bevel-tip needle.

Chapter 3 described the experimental setup that was used to evaluate the algorithms presented in Chapter 2. The experimental setup had a six degrees-of-freedom robotic arm that was used to perform needle manipulations. An electromagnetic tracking system was used to measure needle tip deflection. The Bezier curve trajectory prediction method was able to predict trajectories to within 1.5 mm accuracy, while the adaptive LSTM-SG filter method was able to predict with 3.5 mm accuracy. Both methods had limitations in that they could not accurately or repeatedly make predictions during early stages of the needle insertion where the depth was less than 80 mm. The experimental setup described in Chapter 3 was also used to collect data for the purpose of training deep neural networks in Chapter 4.

Chapter 4 proposed a deep recurrent neural network architecture that could be used for simultaneous path planning in the position domain, while commanding needle manipulations in the velocity domain. This architecture used an LSTM auto-encoder to compress sequences of measurements during a needle-based intervention into a latent representation. A sub-component of the network was trained to map latent representations from (a) the time step prior to a needle being rotated, to (b) the timestep after the needle was rotated. This work demonstrated that fully connected layers in a neural network can be used to transform a latent representation of a sequence of measurements, in the same way as if a known sequence of inputs was applied to the system. The sequence of known manipulations was encoded as a single variable, $\beta$, that represents the amount by which the needle was rotated during that sequence of known manipulations. With enough data in simulation, it was shown that this sub-component of the network enabled a numeric solver to predict how the needle trajectory would evolve in response to the needle being rotated by some angle at its current location. The output of the transformation sub-component was passed through additional layers to produce a needle trajectory prediction horizon. The same network was trained using data collected on the experimental setup discussed in Chapter 3. As expected, the network did not train as well with the experimental data as
with the simulation data. Investigations were done to determine an amount by which the size of the dataset collected experimentally would need to increase such that the performance was comparable to that of the simulation network.

In summary, the main contributions of this thesis are two-fold: (a) Exploring methods of trajectory prediction where the predictions are entirely generated based on observed measurements during each insertion; and (b) demonstrating the use of neural networks to explore reachable trajectories of a needle during needle-based interventions. With the current state of the work as presented in this thesis, the Bezier curve (BC) method is the preferred method. This method does not require the second derivative for the trajectory to be constant as is the case with the SG method; This enables the algorithm to capture macro trends in the second derivative which can likely be attributed to a gradient in stiffness across different layers of the skin. The neural network approach likely has the highest ceiling of performance potential as the properties of the network allow very nonlinear attributes of the needle-tissue interaction to be captured. However, the neural network approach would require the collection of many data points for the model to generalize well to a variety of tissues with different properties.

5.2 Future Work

The main extension of the work presented in this thesis would be to continue gathering data to train the network presented in Chapter 4. To maximize how well the network generalizes, different phantom tissues should be used that have different or varying stiffnesses, while also having abnormalities included within them. Suspending different organs such as liver or kidney within the gelatin would make for a much more robust dataset; however it is expected that the dataset size would need to increase far more than the estimated factor if organs were included in the phantoms. The need for large databases of training data has been recognized and there are active efforts to create such databases. The JIGSAWS database from Johns Hopkins University (JHU) is an example of such a database, where the daVinci® surgical robotic platform by Intuitive Surgical Inc. has been used to collect and label data from a variety of procedures including needle passing.
Improvements could also be made to the experimental setup to reduce misalignments of needle deflection when transforming the needle tip from the robot’s frame to the needle base frame. Smaller EM field generators, such as the ones used in virtual reality gloves [54], could be encapsulated in a new 3D printed end-effector to provide more accurate needle deflection measurements.

Lastly, the loop rates and solver times could be improved if many of the subroutines were written directly in C++ using libraries specifically made for trajectory prediction, as opposed to one of the general numeric solvers provided with the MATLAB optimization toolbox.
References


Chris Morley Curriculum Vitae

Education

M.E.Sc (Robotics & Control) 2020-Current
University of Western Ontario (Western University), London, Ontario, Canada
Thesis: Online Trajectory Generation Strategies for Needle-Based Interventions

B.E.Sc (Mechatronic Systems Engineering) 2016-2020
University of Western Ontario (Western University), London, Ontario, Canada
Graduated with Honors

Skillset

- *Languages:* proficient {MATLAB/Simulink}, competent {C++, Python}
- *MATLAB Toolboxes:* Robotics System, Control System, Deep Learning, Simscape Multibody, Model Predictive Control, Optimization
- *Technical:* Linux, Git, ROS
- *CAD Software:* Creo Parametric, Fusion 360, SolidWorks
- *Build Tools:* Catkin, CMake, QUARC
- Machine Learning (PyTorch, Keras, MATLAB Deep Learning Toolbox)
- Efficient deployment of novel algorithms to hardware (ROS, MATLAB Level 2 C MEX S-Fonctions)
- Modeling of Dynamic Systems and Appropriate Control Algorithms (Model Predictive Control, Sliding Mode, Adaptive, Robust, Impedance, Hybrid Force-Position)
- State Estimation
- Optimization

Conference Papers


Certifications

- Certified SolidWorks Associate (CWSA ID: C-EQ762L888X)
- Certified SolidWorks Associate – Simulation (CWSA-S ID: C-JGR2HBYHK6)

Research Projects

Autonomous Needle Insertion for Deep Tissue Applications 2021-current

- Work focuses on accurately steering brachytherapy/biopsy needle towards the intended target while minimizing total rotation over the course of insertion.
- Automated a 6 degrees-of-freedom (DOF) robotic manipulator to perform autonomous needle insertions on gelatin phantom tissue. Automation enabled large datasets to be collected.
- Recurrent neural network trained on datasets successfully models the dynamics of needle-tissue interaction, allowing for the prediction of future trajectories. The neural network is used as a system model within a model predictive control (MPC) framework to determine an optimal sequence of needle manipulations over a finite horizon.
- Transformations on the latent representation of time series measurements enable MPC to evaluate action sequences as a single action, thus allowing for a longer useful prediction horizon.
- Kalman filtering was used to improve force & position measurement accuracy.
- Custom Simulink library was developed to interface with NDI Aurora electromagnetic tracking system. An electromagnetic tracker is embedded within the needle tip, allowing the deflection of the needle tip to be measured in 3 DOF.

**Analysis of Nonlinear Control Algorithms using Kinova Gen2 Manipulator** 2021
- Equations of motion for the Kinova manipulator were calculated using both Newtonian and Lagrangian Mechanics. The results showed that Newtonian mechanics result in more computationally efficient equations; however Lagrangian mechanics generate a more computationally efficient derivative of the Jacobian matrix which is needed when implementing certain impedance control laws.
- Regressor matrix was derived in order to implement a sliding adaptive control law. This enabled accurate trajectory tracking despite uncertainty in the inertia parameters of the manipulator links.
- A custom MATLAB Level 2 C MEX S-Function was written to communicate with the manipulator via the manufacturer’s API. This function takes care of start-up routines, gravity calibration, and obstacle avoidance while ensuring that appropriate joint commands are provided prior to switching joint control modes (torque, position velocity). A custom GUI was designed and communicated via UDP protocol with a state machine in Simulink. The state machine would send requests and receive responses from the custom S-Function when a change in joint control mode was requested.

**Hearing-Assisted Eyewear (Fourth Year Capstone Project)** 2019-2020
- Project objective was to enhance one’s ability to isolate audio of interest in a noisy environment.
- Responsibility was to implement an algorithm that could track the user’s pupil, estimate their gaze point, and orient an array of microphones towards the target.
- CMOS sensor in combination with an IR light was used to provide visual input of the user’s eye movement. Image processing was performed to localize the center of the pupil in the camera frame.
- Calibration was performed after collecting data from several users to determine a mapping between 2D pixel coordinates of the pupil and the user’s gaze point, described with spherical coordinates.
- Code was written in MATLAB and deployed to a Raspberry Pi using MATLAB Coder.

**Dynamic Systems Modeling, Analysis & Control** 2018-2020
- Several different systems were simulated using Simscape Multibody. These systems include an inverted pendulum, multi-mass spring dampers, a ball balancing table, and 3 & 6 DOF robotic manipulators.
• Equations of motion for all systems were derived using Lagrangian or Newtonian mechanics.
• Different control laws were implemented and compared.
• All these examples are available on GitHub.

Work Experience

Telerobotics Engineer, Sanctuary AI, Vancouver, British Columbia, Canada  Sept 2022-Current
• Implement audio-video pipeline that is feed to VR headset to enable remote operation of humanoid robot.
• Modeling and control of 21 DOF (2x 9 DOF arm + 3 DOF torso) exoskeleton used to control remote humanoid robot.
  o Sensor less state estimation of contact force between operator and distal frame of each arm.
  o Velocity unilateral tracking of second exoskeleton unit in cartesian space with drift compensation.
  o Force-feedback to human operator at distal frame of both arms.
• Self-collision avoidance for 17 DOF robot hand

Lead Product Engineer, AHead Simulations, Cambridge, Ontario, Canada  Summer 2020
• Designed a Nasal Swab Education tool which was used in COVID-19 testing centers to demonstrate the correct procedure when performing a nasal swab.
• Performed rapid prototyping using SLA & FDM 3D printers.
• Investigated different surface finishing methods to improve the transparency of the final product.
• Embedded IR sensor within the product to provide an indirect measurement of the force being applied to the nasal pharynx. LEDs were used to provide visual feedback from the reaction force measured at the nasal pharynx.
• Worked with Ear, Nose and Throat (ENT) Doctor in Middlesex-London Health Unit to ensure the 3D printed anatomy was correct and that the LEDs were calibrated correctly so that visual feedback provided users with meaningful information about the swab they performed.

Teaching Experience

Sensors & Actuators (MSE 3302) – Teaching Assistant  Spring Term 2020/21
• Created interactive Simulink (Simscape) models to replicate the experience students would usually have in the lab. In-person labs were not possible due to COVID-19.
• Held weekly virtual office hours to assist students with course material.
• Gave lectures on best practices for modeling systems using Simulink.

Robotic Manipulators (MSE 4401) – Teaching Assistant  Fall Term 2021
• Ran weekly labs assisting students with concepts such as forward/inverse kinematics & dynamics.
• Marked assignments, lab reports & exams.
• Held review lecture prior to the final exam to help students with course material.