USING HIGH-LEVEL PROCESSING OF LOW-LEVEL SIGNALS TO ACTIVELY ASSIST SURGEONS WITH INTELLIGENT SURGICAL ROBOTS

by

Jie Ying Wu

A dissertation submitted to Johns Hopkins University in conformity
with the requirements for the degree of Doctor of Philosophy

Baltimore, Maryland
August, 2021

Abstract

Robotic surgical systems are increasingly used for minimally-invasive surgeries. As such, there is opportunity for these systems to fundamentally change the way surgeries are performed by becoming intelligent assistants rather than simply acting as the extension of surgeons' arms. As a step towards intelligent assistance, this thesis looks at ways to represent different aspects of robot-assisted surgery (RAS).

We identify three main components: the robot, the surgeon actions, and the patient scene dynamics. Traditional learning algorithms in these domains are predominantly supervised methods. This has several drawbacks. First many of these domains are non-categorical, like how soft-tissue deforms. This makes labeling difficult. Second, surgeries vary greatly. Estimation of the robot state may be affected by how the robot is docked and cable tensions in the instruments. Estimation of the patient anatomy and its dynamics are often inaccurate, and in any case, may change throughout a surgery. To obtain the most accurate information, these aspects must be learned during the procedure. This limits the amount of labeling that could be done. On the surgeon side, different surgeons may perform the same procedure differently and the algorithm should provide personalized estimations for surgeons. All of these considerations motivated the use of self-supervised learning throughout this thesis.

We first build a representation of the robot system. In particular, we looked at learning the dynamics model of the robot. We evaluate the model by using it to estimate forces. Once we can estimate forces in free space, we extend the algorithm to take into account patient-specific interactions, namely with the trocar and the cannula seal. Accounting for surgery-specific interactions is possible because our method does not require additional sensors and can be trained in less than five minutes, including time for data collection.

Next, we use cross-modal training to understand surgeon actions by looking at the bottleneck layer when mapping video to kinematics. This should contain information about the latent space of surgeon-actions, while discarding some medium-specific information about either the video or the kinematics.

Lastly, to understand the patient scene, we start with modeling interactions between a robot instrument and a soft-tissue phantom. Models are often inaccurate due to imprecise material parameters and boundary conditions, particularly in clinical scenarios. Therefore, we add a depth camera to observe deformations to correct the results of simulations. We also introduce a network that learns to simulate soft-tissue deformation from physics simulators in order to speed up the estimation.

We demonstrate that self-supervised learning can be used for understanding each part of RAS. The representations it learns contain information about signals that are not directly measurable. The self-supervised nature of the methods presented in this thesis lends itself well to learning throughout a surgery. With such frameworks, we can overcome some of the main barriers to adopting learning methods in the operating room: the variety in surgery and the difficulty in labeling enough training data for each case.

Thesis Readers

Dr. Peter Kazanzides (Primary Advisor), Research Professor, Johns Hopkins University

Dr. Mathias Unberath (Co-advisor), Assistant Professor, Johns Hopkins University

Dr. Russell H. Taylor, John C. Malone Professor, Johns Hopkins University

Dr. Omid Mohareri, Manager, Intuitive Surgical Inc.

This thesis is dedicated to all the teachers I have had, who always saw more in me than I did.

Acknowledgements

First and foremost, I would like to thank my primary advisor, Prof. Peter Kazanzides, for his endless support, whether it is in keeping lab lunch going or for my sometimes far-flung research ideas. Before I even joined Hopkins, I was told that he is one of the nicest people I will ever meet and he has lived up to that praise. He always challenges me to seek to thoroughly understand a problem and reminds me that my methods should address real challenges.

I would also like to thank my co-advisor Prof. Mathias Unberath for guidance in both research and professional development. He supported my transition to add more machine learning in my research and I appreciated the opportunity to be part of developing the deep learning curriculum. I would also like to thank the rest of my thesis committee. Thank you Prof. Russell Taylor for all the interesting discussions about my research at lab meetings. Dr. Omid Mohareri, thank you for mentoring my first industry experience and being my sounding board for my research direction since.

I would like to thank Prof. Mehran Armand for introducing me to Hopkins through hosting me as a summer researcher before I went off on my gap year, and serving on my GBO committee. I would like to thank the other members of my GBO committee, Prof. Noah Cowan and Prof. Ryan Huang. I would like to thank all my clinical collaborators, Jacqueline Kikuchi, Grace Chen, Greg Osgood, and Alex Johnson, for reminding me as an engineer what is truly important.

To Anton Deguet, I appreciate how you would not only be available to help me run the robot, but if you walked by my desk and saw me stuck on some program build, you would always stop and fix it. To Prof. Nassir Navab, thank you for welcoming me to your group both in Baltimore and in Munich. To Iulian Iordachita, thank you for your guidance in designing phantoms and your insightful questions at lab meetings.

To all my colleagues in all the labs I had been in, thank you for making my PhD experience. Thank you Dr. Javad Fotouhi and Daniil Pakhomov for always keeping our office pod interesting. In the SMARTS lab, thank you Dr. Zihan Chen and Dr. Long Qian for your invaluable advice on the dVRK and AR, and more generally on the PhD program and life. Thank you Dr. Ehsan Azimi for helping me cram for my GBO. Thank you Will Pryor for being a great housemate and all the board game nights. Thank you Keshuai Xu for sharing all your electronics expertise and letting me dog sit. Thank you Dr. Adnan Munawar for making my projects look pretty. To Dayeon Kim and Nick Greene, thanks for joining me these past couple years.

To Benjamin D. Killeen, Kinjal Shah, Anna Zapaishchykova, Philipp Nikutta, Aniruddha Tamhane, Shreya Chakraborty, Jinchi Wei, Tiger Gao, Mareike Thies, thank you all for diving into the COVID dataset project. I still cannot believe what we pulled off there. To the rest of ARCADE lab, Baichuan Jiang, Catalina Gomez, Cong Gao, Gabe Villasana, Hao Ding, Haomin Chen, Max Li, Nathan Drenkow, Weiyao Wang, Wenhao Gu, Xingtong Liu, Yicheng Hu, Yue Fan, thank you for letting me moonlight in your meetings and discussions. To the BIGSS lab, especially Dr. Robert Grupp, Dr. Shahriar Sefati, Prof. Farshid Alambeigi, Dr. Amirhossein Farvardin, Joshua Liu, and Dr. Ryan Murphy, thank you for being my first home at Hopkins.

Thank you Dr. Simon DiMaio, Dale Bergman, Kollin Tierling, and Ted Walker for hosting me at Intuitive and the continued collaboration. Thank you to my collaborators on various projects: Nural Yilmaz, Nam Tran, Ugur Tumerdem, Dou Qi, Laura Fink, Yong-Hao Long, Bo Lu, Yue-Min Jin, Tianyu Song, Yordanka Velikova, Yun-Hui Liu, Peng-Ann Heng, Marc Stamminger, and Jintan Zhang.

To Molly O'Brien, thank you for exploring Baltimore and figuring out this PhD thing with me. To Sing Chun Lee, thank you for introducing me to all sorts of restaurants and theatres. To Jalaj Upadhyay, thank you for reminding me that Canada is the best. To my ex-housemates Alex Barthel, thank you for all the adventures. To my college roommate Dr. Julia Bu, thanks for all the story times and life-debugging sessions. To my college roommate Dr. Theresa Cloutier, thanks for keeping me sane on an hourly basis and for being my first reader on everything. To Christopher Schlichter, thanks for reminding me that there is a life outside of academia.

Finally, I would like to thank my parents, Qiong Shao and Jun Ye Wu, for their constant love, guidance, encouragement, and support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Robot-assisted minimally invasive surgery is a fast-growing market

Minimally-invasive surgery (MIS) is becoming an increasingly viable alternative to open-surgery, especially for laparoscopic procedures in soft tissue. It reduces risks to the patient and improves recovery time [1, 2]; however, MIS is often more technically challenging for the surgeon and may lead to increased intraoperative complications [3]. To ease the adoption of laparoscopic procedures, many hospitals have acquired robotic surgical systems, mostly in the form of a da Vinci Surgical System (Intuitive Surgical Inc., Sunnyvale, CA) [4]. To date, more than 6000 da Vinci systems have been installed in hospitals around the world.

In a Strengths, Weaknesses, Opportunities, and Threats (SWOT) analysis, Dao et al. [5] identified that da Vinci surgeries are correlated with less trauma to the patient, in terms of lower complication rates, shorter length of hospital stays, faster recovery, less blood loss, and lower risk of infection compared to open surgery. These systems also help surgeons by providing higher precision through tremor reduction and motion scaling, improving the learning curve compared to laparoscopic surgery, and enabling better ergonomics. The authors also state that there is potential for improvement such as by including more artificial intelligence (AI) and connected devices, such as in the internet of things.

Using AI could expand the capabilities of the robotic system to be more than just an extension of the surgeons' arms. One example for how AI can help would be to provide additional anatomy information. Placing endoscopic scenes in context of preoperative scans remains a major challenge for surgeons. Since anatomy placement can shift drastically, surgeons generally require many years of experience to learn it. Surgical systems could ease this challenge by providing relevant guidance to surgeons based on a preoperative plan, intraoperative sensing, and an understanding of the current state of surgery. If the computer has a good model of the anatomy, it could identify where arteries are in the preoperative model, register them to the scene the surgeon is operating in, and warn the surgeons if they get too close. If the computer has a good model of the surgeon's actions, it could provide guidance, such as by shifting the camera view to better see the target. If it can combine both a modeling of the surgical scene and the surgeon's actions within that scene, it could take over repetitive subtasks like suturing to reduce surgeon fatigue.

This thesis explores how we can use machine learning to build these models to support clinical procedures with surgical robotic systems. It examines the use of self-supervision to 1) interpret low-level signals into clinically relevant information, 2) build a representation of surgeon actions, and 3) create a better representation of the surgical scene by adding additional sensors. The goal of the thesis is to build representations of the robot-assisted surgery (RAS) as a step towards making the robot an intelligent assistant to the surgeon, rather than just an extension of the surgeon's arms.

Adding intelligence to surgical robots is increasingly relevant as more hospitals adopt robotic systems. As an example of growth in RAS, Fig. 1-1 shows the percentage of open vs. laparoscopic vs. robotic surgeries over a 6 year period in 73 hospitals in Michigan. While the percentage of procedures done as open-surgery decreases throughout the study period, the laparoscopic percentage initially increases. The study attributes the decrease after 2016 to more hospitals acquiring robotic systems, corresponding to the increase in slope of the robotic cases.

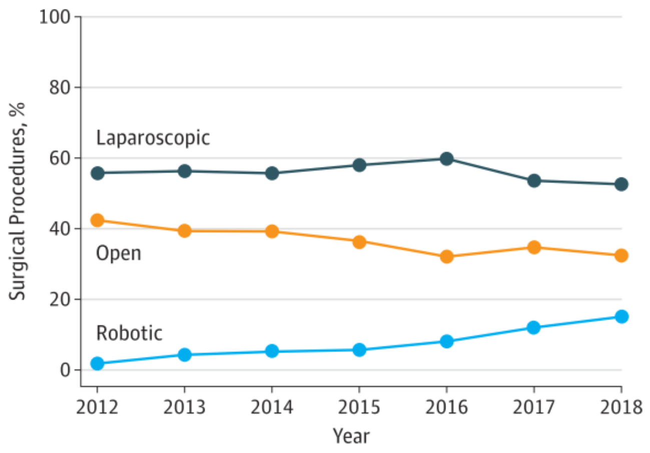


Figure 1-1. Trends in surgery type over 2012-2018 in 73 Michigan Hospitals. Study cohort included 169 404 patients [4]

1.1.1 Technical proficiency is a key indicator of patient outcome

Unfortunately, in manual laparoscopic surgery, patient outcome depends greatly on surgeon skill [6]. RAS has been shown to be more accessible and help novice surgeons perform tasks more easily [7]. However, how to train surgeons for RAS remains an open question [8]. There are currently no standard metrics for acquired skills [9] and surgeons may lose skills quickly [10]. On one hand, these systems have allowed more surgeons to be able to perform these technically challenging laparoscopic procedures. On the other hand, they have not been shown to consistently improve patient outcomes compared to manual laparoscopic surgery so may offer limited benefits to experienced surgeons.

Currently, robotic surgical systems function as a sophisticated extension of the surgeon's arms and improve ergonomics. This is good for regulatory purposes since they essentially function as laparoscopic instruments, but does not take advantage of all the ways such a robot system may aid the surgeon. By adding more sophisticated signal processing, these surgical systems may provide intelligent assistance and further improve safety and efficiency of these systems. This may allow RAS to go beyond the benefits provided by laparoscopic surgery.

1.2 Intelligent assistance can augment surgeons' abilities and reduce dependence on skill

Intelligent assistance has been proposed to improve the safety and reliability of RAS [11]. Such a system could provide additional context for surgeons such as anatomical landmarks based on preoperative imaging, prevent surgeons from accidentally going into a critical area, and provide continuous monitoring of potential negative outcomes. In addition to supporting surgeons throughout a procedure, continuous monitoring could provide personal feedback to surgeons on their skills and highlight areas for improvement to bridge the gap identified in skill level of surgeons [12]. An intelligent assistant can make RAS easier to perform and reduce dependence on surgical skill. Yang et al. proposed a six-levels of autonomy scheme from pure manual control to pure autonomy [13]. Fig. 1-2 shows the gradual shift between human-control and robot-control for each level.

The deformable nature of soft-tissue complicates automated scene modeling and makes it hard to autonomously carry out procedures. Currently, no commercial system performs at any level of autonomy in soft-tissue procedures, but systems up to level three are used in specialties where bones provide structure to the surgical scene, such as in orthopedics. In 1992, ROBODOC [14][15][16][17] conducted a pilot study to perform automated total hip replacement [18]. It used three bone screws placed into the patient's femur to align the pre-operative plan with the surgical scene. Since the scene is rigid and patient is externally fixed, the robot can build an model of the scene and drill automatically, following the preoperative plan. Cyberknife, introduced in

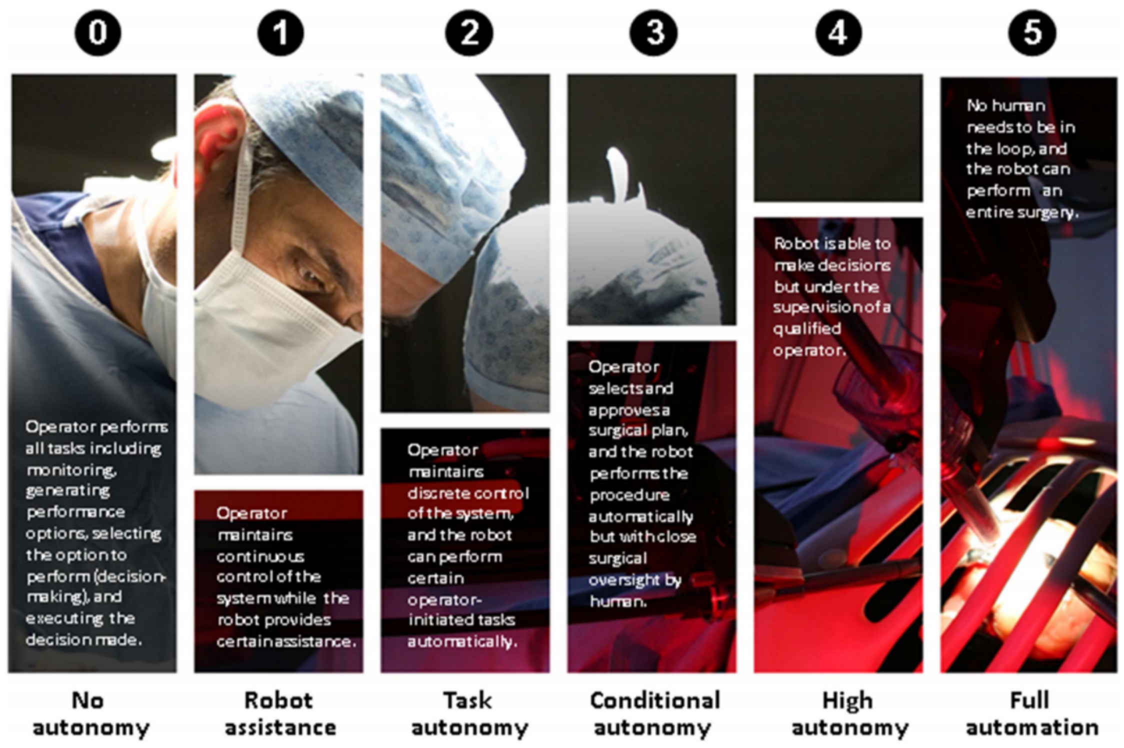

0
1
2
3
4
5
Operator performs all tasks including monitoring, generating performance options, selecting the option to perform (decision-making), and executing the decision made.
Operator maintains continuous control of the system while the robot provides certain assistance.
Operator maintains discrete control of the system, and the robot can perform certain operator-initiated tasks automatically.
Operator selects and approves a surgical plan, and the robot performs the procedure automatically but with close surgical oversight by human.
Robot is able to make decisions but under the supervision of a qualified operator.
No human needs to be in the loop, and the robot can perform an entire surgery.
No autonomy
Robot assistance
Task autonomy
Conditional autonomy
High autonomy
Full automation

Figure 1-2. Proposed levels of autonomy [13]. Currently, teleoperation systems for soft-tissue procedures are at level 0 or 1. Robots operating at up to level 3 have been used in other fields such as orthopedics and ophthalmology, where the environment is better defined.

1997, is a standard tool for cancer treatment [19]. A doctor delineates the area to be targeted and the robot uses preoperative images and sensing to deliver the correct radiation dose to that area. Ophthalmology is another field where RAS has achieved level three autonomy. In both cataract and LASIK surgery, a robot controls the laser and cuts according to a surgical plan created by the surgeon. Since the patient may be awake during these procedures, the laser path must account for eye movement. This is infeasible for the surgeon, but the robot is capable of tracking and reacting at a much faster rate. To limit effects of potential errors in the robot behaviour, the surgeon retains control over the laser beam and can toggle it on and off with a foot pedal while the robot drives the motion.

The problem is comparatively simpler in these situations than general surgery since the areas of interest are surrounded by bones. Bones provide structure for the robot to map the preoperative plan onto. Additionally, markers can be attached to register between the preoperative imaging and the surgical scene. This is more challenging in soft-tissue surgeries where there is no such scaffolding. Given the deformable nature of the anatomy, there is more work to be done in modeling the scene. This thesis explores ways to build an understanding of soft-tissue procedures to carry these ideas into that domain.

1.2.1 Intelligent assistance depends on accurate models of surgeries

Some previous works overcome the challenges of modeling soft-tissue movement by placing many sensors in the environment. In order to perform automated suturing, the Smart Tissue Anastomosis Robot (STAR) combines markers, 3D camera, force sensors, and requires submillimeter precision in tool position [20]. They showed that using the automated system for anastomosis led to more even suture placement though at the cost of longer surgery time compared to RAS and open surgery. The automated system is comparable to the time it takes to perform the procedure laparoscopically.

A stereo camera may provide sufficient information about the scene deformations given knowledge of where the robot instruments are. The SuPer framework combines instrument tracking and a particle filter approach to track tissue deformations from stereo images rather than requiring the additional sensors [21].

Using more sensors vs. a stronger model presents the trade-off between increasing complexity in setting up a surgery vs. making assumptions on how the tissue can deform. Since most cameras only measure surface deformations or marker positions, interpolating how internal deformations occur remains unsolved. Models of the anatomy may be used to estimate deformations but they may be hard to obtain for specific patients. General atlas models are not accurate enough for fully automated procedures.

1.2.2 Learning models from data

An alternative to strong modeling assumptions is to learn these models from data. Murali et al. showed that robots can learn to repeat surgical subtasks by observing human demonstrations [22]. They train a state machine with a vision processing unit to observe whether tasks were successful for choosing transitions. The drawback here is rigid states and transitions do not generalize well to unstructured surgical environments.

Deep learning has emerged in recent years as a powerful tool to model many problems that are hard to mathematically derive. Instead of seeking the best model, deep learning uses high capacity but simple models to learn arbitrary functions from data. As such, it requires much more data than other machine learning algorithms up to this point and, more importantly, the relationship it builds between the input and output is often ill-understood. Both these points have slowed the adoption of deep learning techniques in the medical domain, where patient privacy often limits the amount of data one can collect and surgeons require interpretable models to place trust in the algorithms.

One method to overcome some challenges is to move away from explicit labels in the form of self-supervised learning. This could be done in videos by exploiting synchronicity of signals, such as through the correlation between the sounds and images [23]. In RAS, this synchronization could come in the form of synchronizing different signals measured by the robot.

By incorporating both model-based calculations and machine learning, we can simplify the function the machine learning model needs to learn, and bound the potential errors it can produce. A method such as SuPer Deep [24] extends previous works in [21] by using deep learning for identifying the instruments, but keeps a classical modeling method for tracking tissue deformation. The model could be combined with augmented reality (AR) to provide intraoperative guidance to surgeons such as shown in Fig. 1-3.

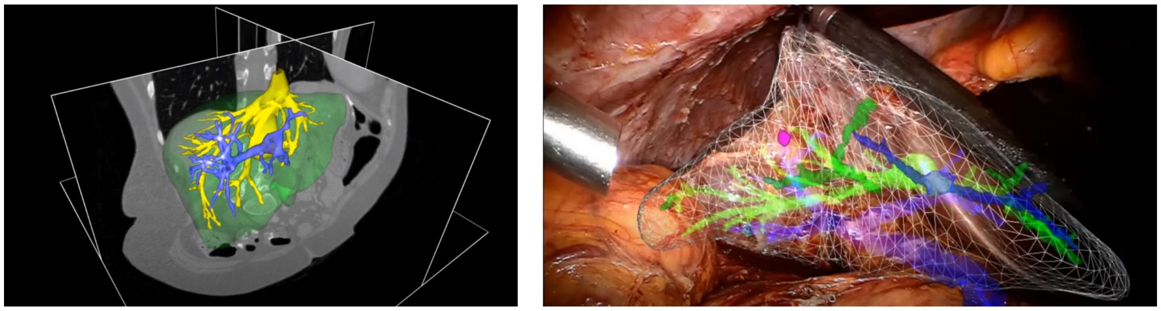


Figure 1-3. Example of how soft-tissue models could be used for intraoperative guidance in robot-assisted surgery from [25]. Critical structures in the liver are identified from a preoperative scan of the patient and a model is created of the organ and its internal structures. The model is registered to the operating scene to show internal structures.

1.3 Learning representations of robot-assisted surgery

Up to now, few works have constructed a holistic representation of a RAS. Instead, works focus on exploring a specific component, which this thesis groups into three sections: the robot, the surgeon, and the scene. This thesis seeks to construct a representation for each of these parts. While each of the representations are validated by targeted experiments, this thesis aims to move away from narrow methods that solve a particular problem towards a more holistic representation of surgery.

It is only by interpreting all of these components that a robot can intelligently aid the surgeon. The robot requires a model of both the surgeon's intent and the scene to autonomously perform actions that would benefit the surgery. One use case is to drive a third arm to keep tissue out of the field of view. For example, in a phantom hysterectomy study conducted at Hopkins, the surgeon switched between cutting and manipulating the uterus. Fig. 1-5 shows an example where the third arm is not actively controlled and is used to keep the uterus out of the way. The other two arms were controlled by the surgeon for tensioning and cutting. Changing the field of view

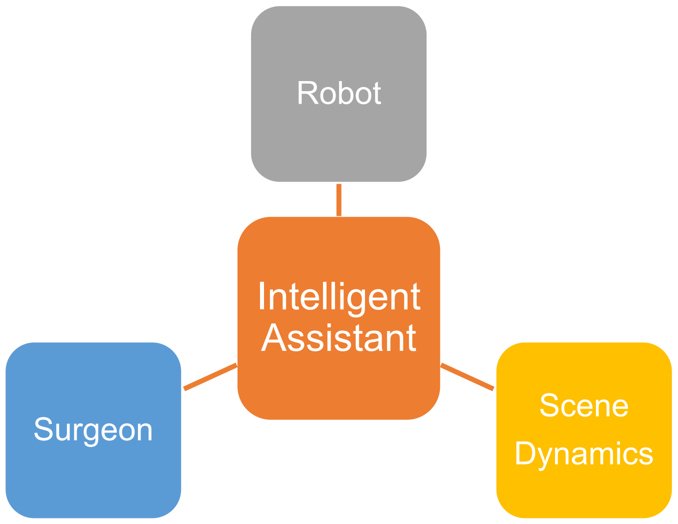


Figure 1-4. We break down the intelligent assistant into three components.

required adjusting the positioning of the organ, and thus switching the active arm. The uterus also slipped occasionally from the third arm's grasp to block the view. In that case, the surgeon had to pause her actions and switch back to controlling the third arm. As the third arm is not always in the field of view, this occasionally required lengthy camera adjustments as well as switching between instruments.

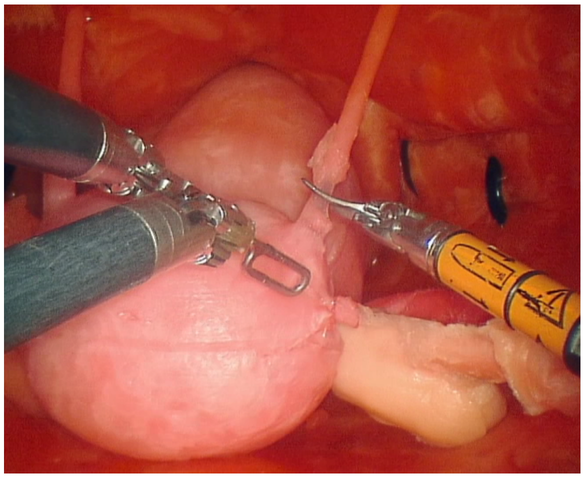


Figure 1-5. Hysterectomy on a phantom. The two arms in the foreground are controlled by the surgeon. The left one is creating tension while the right one is getting ready to cut. The third arm is in the back, keeping the uterus in place.

If the robot could interpret the surgeon's intent, namely to keep a clear field of view for the incision, it could use its model of the soft tissue dynamics to automatically drive the third arm. Furthermore, it could use its sensor readings to detect the force exerted on the instrument by the uterus. If this force changes, this could indicate the organ is slipping out of the instrument's grasp and requires readjustment of the grip. This would allow the surgeon to focus on the primary task: making the incision.

Another use case is to provide guidance to surgeons when they are targeting lesions that are not immediately visible on the surface. Fig. 1-3 shows how a model of critical structures and a lesion could be constructed from a preoperative scan. The right image shows how the model can be combined with AR to provide better guidance to the surgeon. Currently, this model is deformed intraoperatively by registering the model and organ surfaces. This limits guarantees one could make on how the underlying structures might have shifted.

Instead, if we start from a soft-tissue model, we could use the kinematics and the forces, as estimated by our robot model, to estimate the model's deformation throughout a surgery. This requires an accurate and fast deformable soft-tissue simulator. The simulator should also account for the tissue property changes that could come from cutting and cauterization. Using this simulator, we could provide similar guidance to that seen in Fig. 1-3 without using deformable registration. Given a sufficiently accurate model, a rigid deformation could match the model and the camera view. This could maintain bounds on errors of estimates of the internal deformations. It could also signal potential errors in the algorithm if the surfaces do not match. In addition to providing guidance to surgeons, the model could bridge the gap between the level of autonomy available in orthopedic procedures vs. soft-tissue.

While this thesis presents the three parts separately, each part can be used to inform the other. For example, the force estimation presented in Chapter 2 could be used as an input for the skill assessment method presented in Chapter 3. It could also be used in conjunction with the tissue simulation in Chapter 4 to check whether the force we expect to measure based on our simulator matches the force the robot actually measures in interaction with the real scene. Using the simulation from Chapter 4 to provide anatomy context may also improve surgeon action estimation in Chapter 3.

1.3.1 Interpreting robot measurements

The robot measures many more signals at a higher frequency than humans are capable of interpreting. These low-level and high frequency signals are often used by internal algorithms to provide smooth operation and controls. While there has been a lot of work in signal processing to enable smooth controls, fewer machine learning tasks have made use of these signals. Instead, machine learning algorithms have mostly used the endoscope and kinematics data from the robot, which have a more obvious link to the task at hand. This in turn makes the training path clearer for learning from how humans perform a task.

Machine learning algorithms do not have the same limitations of human perception though, and this section of the thesis looks at how we could interpret low-level signals to clinically relevant information. It combines low-level signal processing with high-level perception. It uses the example of learning force feedback by learning a robot dynamics model from the relationship between joint motion and torque. We compare performance of two ways to approach such a transformation: the direct way of learning the force from a sensor in the environment, and the indirect way of learning how much torque it takes to drive the robot and subtracting that from measured torque. Chapter 2 further explores how one could adapt the method for use in patients, and update the model throughout a surgery.

1.3.2 Interpreting surgeon actions

Another important aspect of providing intelligent assistance is to interpret what the surgeon is doing and what they are trying to achieve. The actions and guidance the robot provides should be targeted to the surgeon's current task. Previous works have examined how to break up surgical tasks into small, distinct actions, called surgemes [26]. Followup work used future-prediction as a self-supervisory signal to learn representations of the task [27].

The prospect of achieving self-supervised continual learning in surgical robotics is exciting as it may enable lifelong learning that adapts to different surgeons and cases, ultimately leading to a more general representation of surgical processes. Chapter 3 explores a learning paradigm using synchronous video and kinematics from robot-mediated surgery. While the representations are useful immediately for a variety of tasks, the self-supervised learning paradigm may enable research in lifelong and user-specific learning.

1.3.3 Interpreting scene dynamics

Surgical simulations play an increasingly important role for both surgeon education and to aid the surgeon's understanding of the scene [25]. In RAS, we generally have an idea of the anatomy shape from pre-operative imaging; however, the tissue characteristics, in particular the dynamics, are less understood. Systems such as the ROBODOC show that if we have rigid objects, our models of the scene are sufficient for autonomous motion but there remains a gap in our modeling of soft-tissue motion. To model soft-tissue anatomy deformations, Finite Element Method (FEM) simulations have been held as the gold standard for calculating accurate soft-tissue deformation. Unfortunately, they are too slow for intraoperative use and their accuracy is highly dependent on the material properties and boundary conditions, which can be difficult to obtain.

The goal of Chapter 4 is to build a representation of the scene using machine learning. The representation could flexibly learn from both models and real-time feedback. It provides a path towards real-time guidance during surgery that is biomechanics-driven, and can learn throughout a surgery to account for patient-specific characteristics.

1.3.4 The da Vinci Research Kit

The algorithms presented in this thesis may be adapted for a general class of surgery, but for this thesis, I limited the scope of validation to minimally invasive laparoscopic surgery on the da Vinci Surgical System (Intuitive Surgical Inc., Sunnyvale, CA). I collected all the data on the da Vinci Research Kit (dVRK) [28], which combines the mechanical parts of the first generation da Vinci Surgical System with open-source controllers. At the time of writing, this system is used at 39 sites around the world.

The dVRK has two main components: a surgeon's console and a patient side console. The surgeon's console contains two Master Tool Manipulators (MTM), providing 7 degrees of freedom (DOF) for tool manipulation (translation, rotation, and gripper open/close). On the patient side, there are two or more Patient Side Manipulators (PSM), controlled by the surgeon through the MTMs.

1.4 Thesis statement

This thesis explores building representations of the robot interaction, surgeon intent and tissue dynamics in RAS by using machine learning to interpret low-level signals into clinically relevant information. It presents a step towards a holistic representation of RAS that has the potential to adapt to new users and surgeries.

1.5 Thesis outline and contributions

Chapter 2 presents work on interpreting signals from the robot. It focuses on using existing measurements to learn a robot model and using the robot model to provide force feedback. Its contributions are:

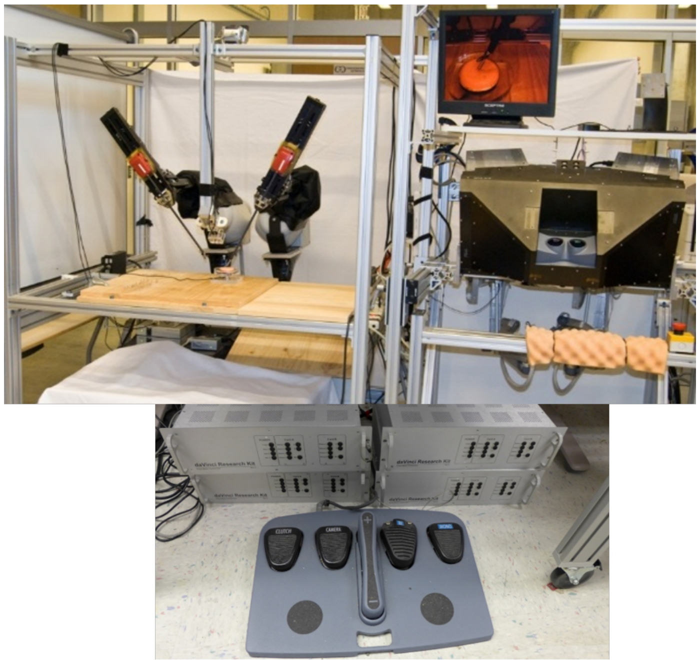


Figure 1-6. The da Vinci Research Kit. The top left and right images show the Patient Side Manipulators (PSM), and the Master Tool Manipulators (MTM) respectively. The bottom image shows the foot pedals and the open-source controller boxes for the system.

1. Direct force estimation method by learning from a force sensor in the environment
2. Indirect force estimation method by learning joint torques in free space motion
3. Demonstration of utility of estimated force feedback for palpation task
4. Two-step training process to adapt to surgery-specific conditions (trocar interactions after docking the robot)
5. Self-supervised method for six DOF estimates of Cartesian forces and torques

Chapter 3 presents work on interpreting signals from the user. By looking at different signals that result from user action, it presents a representation of user actions that is not tied to hand-annotated labels. Its contributions are:

1. Cross-modal, self-supervised network to learn surgical gesture representations
2. Demonstration that the representation captures semantic separations in skill and gesture despite never having been trained with labels

3. Preliminary results that the representations are meaningful even on unseen tasks

Chapter 4 presents work on interpreting the surgical scene. As a geometric model is generally available from preoperative imaging, this chapter focuses on using intraoperative feedback to update dynamics of the model and considers using a learning framework to achieve real-time simulations. Its contributions are:

1. A correction network for finite element method simulations based on real time feedback

2. A network for real time soft tissue simulations

3. Evaluation on gel phantom using the dVRK

Chapter 5 concludes the thesis and details a path towards combining these ideas into an intelligent assistant. By unifying the self-supervised representations presented in the thesis, surgical robots could support doctors in ways that adapt to each surgeon, procedure, and patient.

Chapter 2

Interpreting the robot

In this chapter, we seek to interpret low-level signals from the robot as clinically relevant signals. We seek to implicitly learn a robot model and focus on the application of using it to sense forces. Real-time force sensing is a desired capability in RAS to enable haptic feedback for surgeons. It potentially helps to further decrease trauma during and after the operation by preventing surgeons from unknowingly applying excessive force on tissue. The benefits brought by the current robotic surgical systems come at the cost that the surgeon completely loses the sense of touch [29]. This had already been compromised by the shift from open surgery, where the surgeon's hands could directly palpate tissue, to laparoscopic surgery, where forces were transmitted to the surgeon's hands via the instruments. Experienced robotic surgeons learned to estimate forces through other cues, such as the tautness of suture or the discoloration of tissue being stretched, but it is widely believed that surgical performance would be improved by the addition of haptic feedback.

There are several challenges to achieving haptic feedback in a telesurgical system. First, it is difficult to integrate force sensors on the instrument tips, especially considering that the instrument must survive several cycles of cleaning and sterilization. Installing a force sensor in the non-sterile part of the robot (i.e., above the instrument) is also challenging because it would require good force estimation algorithms that account for the highly nonlinear cable-driven design of the instruments.

There are also practical difficulties in implementing and testing solutions because commercial systems such as the da Vinci do not allow researchers to directly control the robot arms. Fortunately, the widespread adoption of open research platforms such as the da Vinci Research Kit (dVRK) [28], and the Raven II robot [30] enables researchers to implement and test different controllers, and ultimately share working solutions with the community.

Contributions

This chapter presents two methods for force estimation. The first way estimates forces directly by learning from a sensor placed in the environment [31]. The second way estimates free motion torque and uses that to calculate forces [32]. My role in developing both of these methods was mainly in coming up with the framework, setting up the robot for data collection, and creating scripts for data collection and processing. I mentored visiting students, Nam Tran and Nural Yilmaz, who implemented and tested each method respectively.

Building on the self-supervised learning in the second method, Section 2.5 presents a pathway to account for surgery-specific forces [33]. Existing methods are generally trained on benchtop setups that cannot be personalized to a specific surgery. We demonstrate how intraoperative measurements can be used to improve force estimation in the operating room with no additional equipment. We use a correction network to adapt to intraoperative conditions. Nural Yilmaz designed the 3D printed object for measuring torques. The code for all parts of this chapter can be found at `https://github.com/JieYingWu/dvrk_force_estimation`.

2.1 Related works

2.1.1 Force feedback from added sensor

The lack of haptic feedback in robotic surgical systems has led to a significant amount of research to restore it. Some researchers adopt external feedback by using a miniature force sensor to provide force estimation [34–36]. These implementations require either creating a customized tool tip or attaching the force sensor on an existing tool tip/camera. This adds another layer of complexity to the system and raises questions about cost, sterilizability and compatibility with other medical devices [37]. Researchers at DLR have developed a small 6-axis force sensor placed at the tip of the MIRO surgical robot [38]. Berkelman et al. [39] and Siebold et al. [40] developed three and six axis miniature force sensors, respectively, and attached them to the tips of specially developed surgical instruments. Due to the difficulties in manufacturing such sensors, other researchers have attached flexible capacitive [41], strain gauge based [42], or fiber-optics based [43] sensors on surgical grippers. However, the force/torque sensing degrees of freedom are limited in these approaches. Shahzada et al. place four fiber Bragg grating sensors inside the shaft of the instrument to measure lateral forces [44]. Since these do not measure force in the insertion direction, they show limited benefit for palpation tasks. Intra-corporeal sensor based approaches often present difficulties in manufacturing small, accurate, sterilizable, and robust multi-degrees of freedom (DOF) sensors [45]. These approaches that require modifications to the instrument structure also pose limitations on the functionality and may not be applicable under all operational settings.

To overcome these problems, some researchers have proposed the use of force sensors placed at the noninvasive (extra-corporeal) part of the robot. It is possible to estimate tip forces through extra-corporeal sensors, but since the robot interacts with the patient body at the trocar these forces would also be registered by the sensor.

Additional consideration is required to estimate the force at the instrument tip. In order to filter these forces, the “overcoat” method was proposed by Shimachi et al. [46] by introducing a second trocar sensor. This method has been further developed by Willaert et al. [45] and Schwalb et al. [47]. Since these methods require bulky force sensors placed at the trocar, Fontanelli et al. consider the use of fiber optic sensors in the trocar [48]. These methods can provide accurate estimates of forces in the Cartesian XYZ directions of the tool tip. However, the wrist bending torques cannot be estimated/measured. Furthermore, the placement of force/torque sensors in the trocar may not always be clinically feasible.

2.1.2 Learning to estimate forces from robot state or visual feedback

Since deploying force sensors complicates instrument design and sterilization, another approach is to use force sensors to train a model, which then estimates forces during the operation. Such a setup has been used in [49] to estimate the grasping forces on a custom da Vinci gripper from joint states. The proposed direct method is similar to this approach but estimates interaction forces rather than grip force. Guo et al. [50] trained deep neural networks to estimate gripper forces on 1 DOF instruments, by using joint and force sensor measurements in training. Yu et al. [51] use a similar approach for Cartesian force estimation in three axes.

Other approaches based on external feedback involve extracting visual cues from tissue deformation and using computer vision to estimate forces [52–57]. These demonstrate high accuracy in their setup and show the potential of neural networks and computer vision algorithms to extract force data; however, since they rely on visual features, they may not generalize well to new tissues with different appearances. Also, many of these implementations are only shown to estimate force with a single component (usually Z-axis). In [58], interaction forces are estimated using deep neural networks and external camera images as the inputs with a force sensor used to provide the ground truth.

2.1.3 Modeling cable-tendons

In order to obtain force measurements without additional sensors, force estimation methods can be used. Such approaches have been investigated starting with the first surgical robot prototypes like the Black Falcon [59]. One of the issues observed in the early attempts has been the transmission of robot coupling/internal dynamic forces to the operators in free motion. To solve this issue, in [60], a Coulomb friction compensator was proposed to improve the force estimation results on a customized da Vinci patient side manipulator. In [61], a cable tension estimator was developed to eliminate the effects of cable elasticity in the Raven II system. In [62], a sliding mode perturbation observer was developed for the estimation of grasping force on a customized da Vinci gripper.

To overcome problems with cable-tendon driven surgical manipulators, different actuator/transmission systems and force estimation schemes have also been developed, such as a pneumatic forceps mechanism and pressure based force estimation method [63], a rigid rod driven mechanism with strain gauges on the shafts for force estimation [64], and a rigid transmission system with load cell based estimation [65]. In [66][67], a disturbance observer and neural network based inverse dynamics was used to estimate external forces on a rigid link driven robotic forceps prototype.

With the development of the dVRK, many research groups have started developing dynamic identification and external force estimation methods for the da Vinci systems. In [68], an explicit physics-based dynamic model of the dVRK PSM was developed and parameters of this model were identified together with the free motion torques to estimate external forces under quasi-static external loading. In [69], an LMI method was used for dynamic parameter and joint torque identification of both the MTMs and PSMs without external force estimation. In [70], a linearized model of the PSMs was obtained and the parameters were identified with least squares optimization. External forces/torques were also estimated by filtering out the free motion torques. In [71], an open source convex optimization based toolbox was proposed for dynamic model identification of the dVRK. All of these approaches assume an explicit dynamic model for the system and attempt to identify the parameters of the respective models with the robot following an automated optimal excitation trajectory. Although these model-based approaches have good performance, they do not take trocar and cannula seal interactions into consideration.

2.1.4 Building intrinsic model of the robot

Some researchers use a combination of machine learning and simulation/modeling, making use of a torque observer [72] and neuro-evolutionary fuzzy systems [73]. Yasin and Simaan use a support-vector machine and a least-squares model to measure forces in a high degrees-of-freedom manipulator [74]. In the proposed indirect method, we follow a similar approach with a key difference: the inverse joint space dynamics is identified by black box models in the form of neural networks for each joint, whose complexity can be increased and updated to adapt to various operating conditions. This approach can also help reduce the fitting errors in the explicit model-based approaches.

A similar neural network based approach has been proposed in [75] for a 3-DOF Planar Twin-Pantograph haptic interface. However, unlike our approach, a single neural network has been trained for the robot, and training has been performed with a random persistent excitation trajectory. Abeywardena et al. [76] used neural networks with intrinsic feedback to estimate the grip force in a da Vinci instrument.

2.2 Data collection

We collect all data for this chapter using the dVRK. In these experiments, the da Vinci PSM was unilaterally teleoperated by a human operator with an MTM in freespace and in interaction with a phantom. In this chapter's experiments, the dVRK communicates with the computer through FireWire and the data is captured at 1 kHz. Fig. 2-1 shows a block diagram of the dVRK controller hardware; a more detailed description of the system can be found in [28].

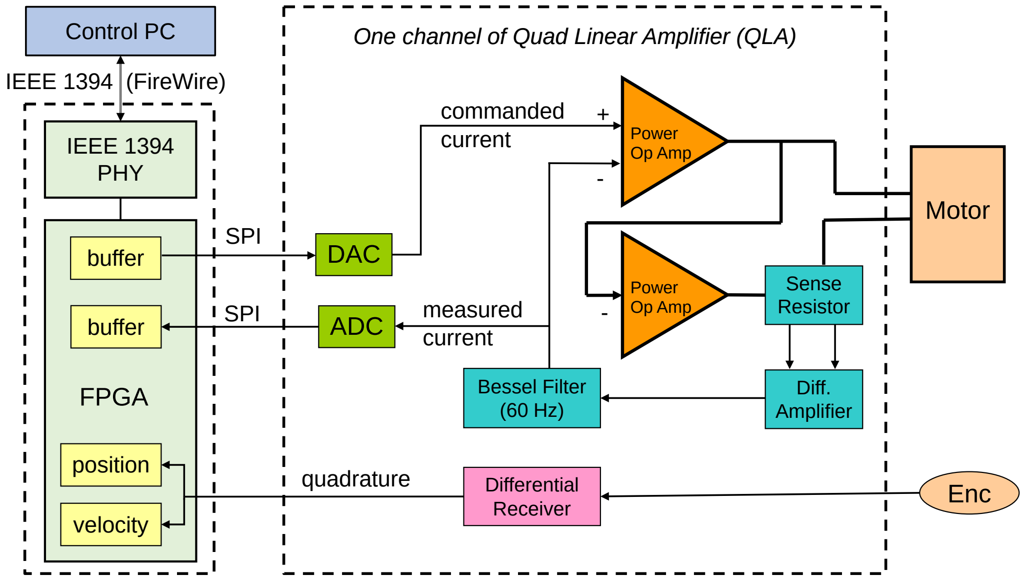


Figure 2-1. dVRK controller hardware for one channel

Robot positions are read from encoders and velocities are estimated from the encoders using the algorithm in [77]. The torque measurement τ is obtained by the multiplication of the measured current values with motor torque/force constants. We capture the data through Rosbags [78]. Within each bag, there is a timestamp for each data value. The dVRKs's sampling rate is set to 1 kHz.

We use a Gamma force/torque sensor (ATI Industrial Automation, Apex, NC) to measure the Cartesian force acting on the phantom. The robot joint state and the force sensor's sampling rate are both set at 1 kHz, though the actual update rate is lower. The actual frequency captured is between 930 to 940 Hz. The force sensor's orientation is manually aligned with the robot world coordinate system. The phantom is placed above the force sensor, secured by a large binder clip. We use linear interpolation to match the timestamps of the robot to the force sensor.

We teleoperate the PSM in both free space motion and to smoothly palpate a phantom mounted on a force sensor. We also include spontaneous movements with varying velocities and contact forces. When gathering no-contact data, we put the PSM in various positions, including near joint limits. All this ensures a diverse range of data for effective learning at as many PSM positions in the workspace as possible. The robot gripper is not used and kept closed during teleoperation as we focused on the palpation task.

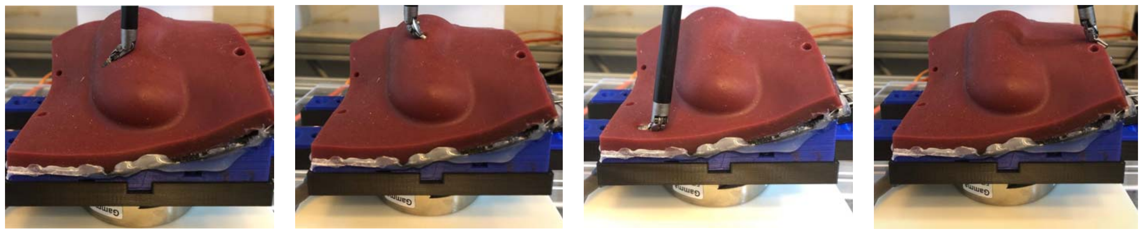


Figure 2-2. Various scenarios on the same phantom are shown during data collection as well as offline evaluation. The force sensor is under the phantom. There is neither gripping nor pulling during teleoperation, as we are more interested in the palpation force with respect to the phantom.

2.3 Direct estimate - learn from force sensor

One approach we took to force estimation is using the network to learn to estimate the external forces given the internal position/velocity and torque measurements [31]. This section proposes a purely data-driven end-to-end force estimation algorithm, based on deep learning, that has the potential to generalize across tissues of different stiffnesses, workspace configurations, and users' behavior. Our deep learning model provides end-to-end force estimation by receiving joint velocities and joint torques as inputs and yielding Cartesian force in the X, Y and Z directions as outputs. We validate our force estimates in a pilot study for a haptic feedback system integrated into the dVRK infrastructure.

Internal dynamics of a surgical robot can be difficult to model. Friction, wear and tear as well as other disturbances within the internal structure of a robot are highly nonlinear. On the other hand, a neural network can be easily retrained with a force sensor and teleoperation data gathered from a new surgical robot. Our method uses signals already measured by the system and does not require additional sensors at runtime (an external force sensor is required when collecting training data).

2.3.1 Method

The neural network is trained by teleoperating the robotic instrument in both free space and in contact with a phantom. A force sensor placed under the phantom provides the ground truth for training. Although using the measured joint position can enable the network to better learn the robot dynamics, such an approach may lead the neural network to learn the position of the phantom. To prevent this, we do not pass the position of the robot to the neural network, and use only the joint velocities, $\dot{q}$, and torques, τ, as inputs to the neural network function, H, to estimate external, Cartesian forces, F_{ext}, as shown in Eq. 2.1,

$$F_{ext} = H(\dot{q}, \tau) \tag{2.1}$$

In our preliminary testing, this worked about as well as an implementation that also used the measured joint positions, which indicates that joint positions have a small effect on the robot dynamics. For instance, the patient side manipulator (PSM) of the da Vinci contains mechanical counterbalances and therefore the effect of gravity is minimal. If necessary, a gravity correction term, such as calculated from [79], can be added to improve the force estimation.

In order to capture some temporal information, we use a window of past joint information as input to the network. The windowing takes into account backlash and hysteresis to make effective predictions for the current time step. We started from a single layer network and empirically increased the network complexity and searched for the optimal hyperparameters. The resulting network has five hidden layers and each hidden layer has 250 neurons, and uses 100 past data points as input. Since each data point consists of six joint velocities and torques, the input dimension is 1200. The output dimension is three, as the ground truth Cartesian force has three components: X, Y and Z. Fig. 2-3 shows the complete data flow of the neural network.

We use Leaky ReLU as the intermediate activation function. The output layer uses the sigmoid activation function and the values are scaled to the measured minimum and maximum detected forces in the training data. We initialize our weights using the He uniform initializer. The loss function is set to MAE (mean absolute error), as when the network is used to provide haptic feedback, transient and large-magnitude errors are not as noticeable as long-lasting and small-magnitude errors.

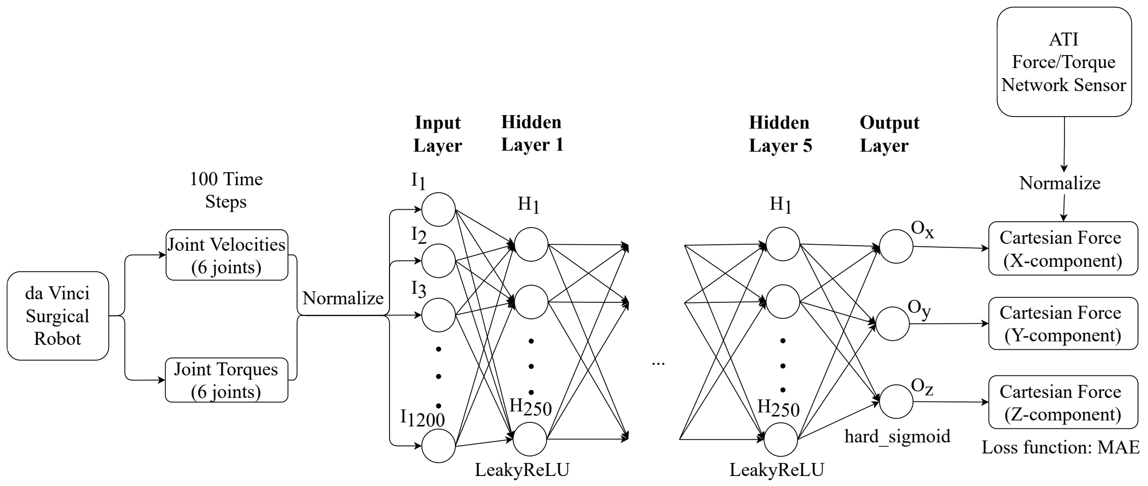


Figure 2-3. Overview of data flow from the robot joint information, through 5 hidden layers of 250 nodes each, and to the output layer. The output is compared to the force sensor readings. The loss function is the mean absolute error.

Implementation

We use the Keras library to implement a fully-connected feedforward neural network, with Tensorflow as backend [80, 81]. We choose SGD over Adam, as it provides more effective generalization in our tests. The learning parameters are set as follows: learning rate: 0.01, decay: 10^{-6} and momentum: 0.9. Nesterov's accelerated gradient descent is used for faster convergence. Table 2-1 summarizes the dataset sizes used in this section and the approximate corresponding duration given the sampling rate of 1 kHz.

Table 2-1. Summary of datasets. The number of raw data points for each set is shown, along with the corresponding duration given the sampling rate of 1 kHz. H: Horizontal Configuration. V: Vertical Configuration

Set	Size	Duration
Training (H)	17,862,662	5.0 hours
Validation (H)	4,465,655	1.2 hours
Offline Evaluation (H)	359,921	5.9 minutes
Offline Evaluation (V)	345,060	5.8 minutes

Fig. 2-4 depicts two sliding window strategies. Method A shifts the window by 100 raw data points, while method B shifts the window by one raw data point. Method A is only used for training, while method B is employed for both offline and online evaluation. We notice that using method B for training yields poorer predictive performance than using method A, possibly since there is high overlap of the samples. Method B provides predictions at the sampling rate of the dVRK, which is particularly necessary during online evaluation in order to provide fast and smooth feedback.

We group every 100 consecutive raw data points into one sample, yielding 1200 features per input. We use the force vector label (containing X, Y and Z components) of the 100th raw data point as the target prediction.

As shown in Fig. 2-5, we collect data in both horizontal and vertical configurations to evaluate our neural network. The representation the network learns is dependent on where we mount the force sensor. We consider it infeasible to mount the sensor

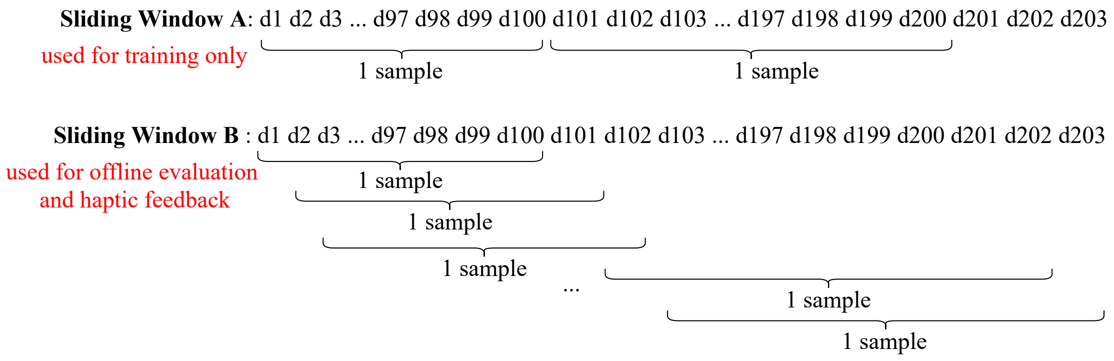

Sliding Window A: d1 d2 d3 ... d97 d98 d99 d100 d101 d102 d103 ... d197 d198 d199 d200 d201 d202 d203
used for training only
1 sample
1 sample
Sliding Window B : d1 d2 d3 ... d97 d98 d99 d100 d101 d102 d103 ... d197 d198 d199 d200 d201 d202 d203
used for offline evaluation
and haptic feedback
1 sample
1 sample
1 sample
...
1 sample
1 sample

Figure 2-4. Sliding window method A (shifting by 100 raw data points) is used for training only, while method B (shifting by 1 raw data point) is used for both offline and online evaluation.

in enough locations to sufficiently capture the six DOF workspace. Instead, we use the vertical configuration to test what range of errors the network would produce in a configuration completely different from the one in which the network was trained. Fig. 2-2 demonstrates our data collection method, in which we ensure a wide range of no-contact and contact scenarios, such as poking with the tip or edge of the gripper. The phantom is the same as the one used in training.

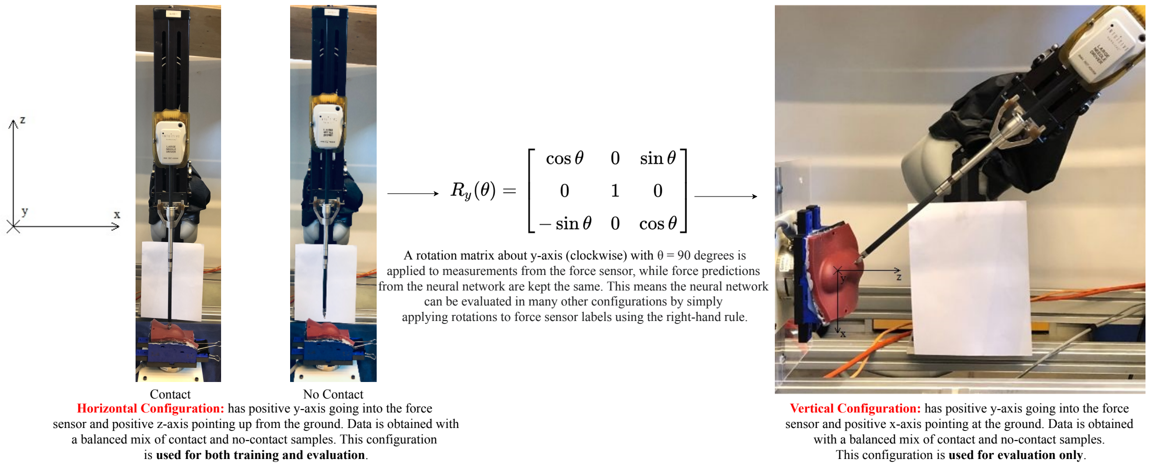

z
y
x
Contact
No Contact
$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$
A rotation matrix about y-axis (clockwise) with θ = 90 degrees is applied to measurements from the force sensor, while force predictions from the neural network are kept the same. This means the neural network can be evaluated in many other configurations by simply applying rotations to force sensor labels using the right-hand rule.
z
x
Horizontal Configuration: has positive y-axis going into the force sensor and positive z-axis pointing up from the ground. Data is obtained with a balanced mix of contact and no-contact samples. This configuration is **used for both training and evaluation**.
Vertical Configuration: has positive y-axis going into the force sensor and positive x-axis pointing at the ground. Data is obtained with a balanced mix of contact and no-contact samples. This configuration is **used for evaluation only**.

Figure 2-5. dVRK test bench is shown with da Vinci robot arm in both horizontal and vertical configurations. The phantom is always placed directly on the ATI network force/torque sensor, all secured by a large binder clip.

2.3.2 Experiments and results

2.3.2.1 Offline evaluation compared to sensor readings

Fig. 2-6 shows offline evaluation results for both horizontal (left) and vertical (right) configurations. For the vertical configuration, a rotation is applied to the force sensor data so that the readings remain aligned with the robot world coordinate system. The red curves show the error, calculated by subtracting predicted values from measured values. Zoomed insets depicting predicted values (blue) and ground truth values (orange) are used to highlight areas with particularly good and bad alignment, revealing the strengths and weaknesses of our neural network in different intervals.

In the horizontal configuration plots, the prediction curve (blue) closely matches the ground truth curve (orange). Minor errors, in which the network incorrectly predicts a force when there is none, are also detected, as can be seen in the Y-component and Z-component plots of the horizontal configuration and the X-component plot of the vertical configuration. There is also a considerable amount of sudden spikes in the red error curve for both horizontal and vertical configurations. Many of these error spikes do approach the highest magnitudes of force exerted in all three components across horizontal and vertical configurations. Some of these error spikes are due to the neural network completely missing the estimation. In situations where the neural network obtains a fairly precise estimation of the force, a small phase shift or lag in the prediction curve can also cause high-magnitude error spikes. An example of such a lag can be seen in the zoomed inset between 180 s to 210 s in the horizontal configuration's Y-component plot.

Table 2-2 summarizes the offline evaluation results, showing both Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).

We observe that the neural network performs better in the horizontal configuration,

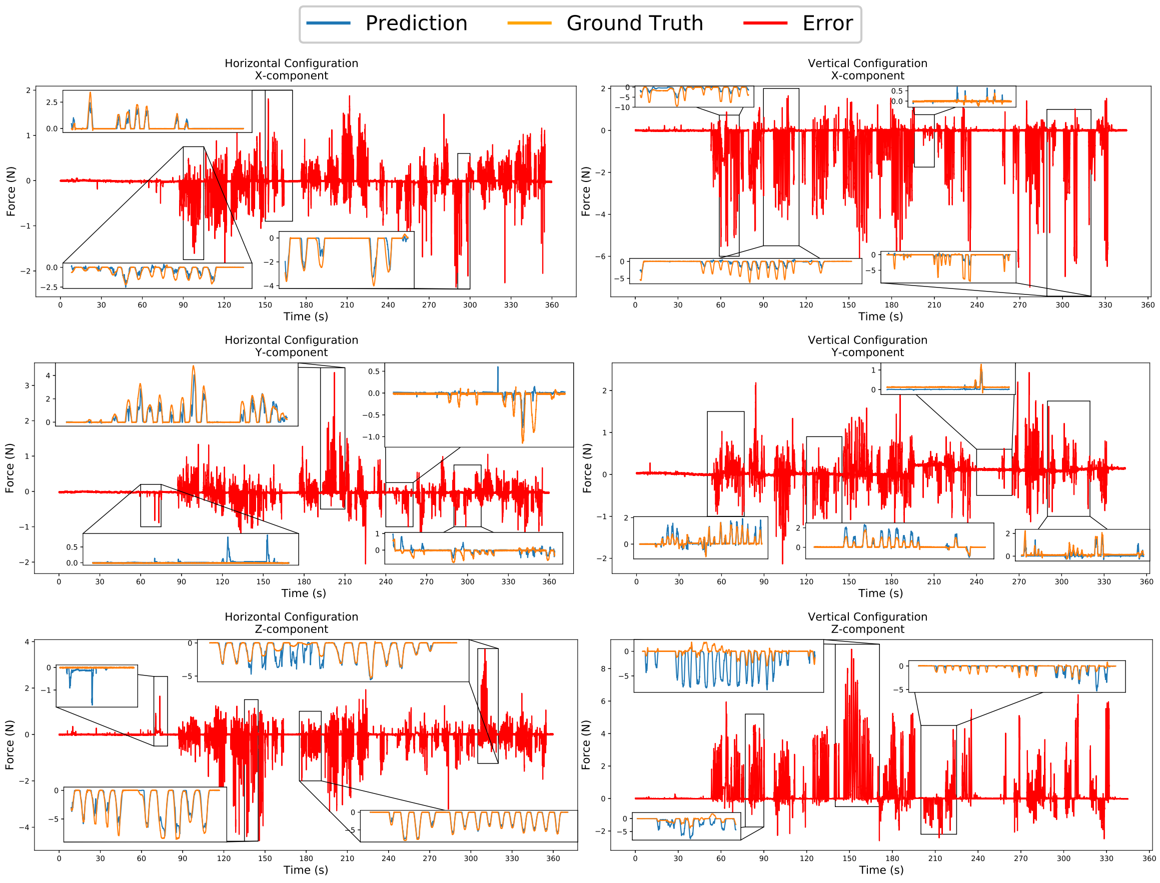


Figure 2-6. Offline evaluation results for horizontal (left) and vertical (right) configurations as depicted in Fig. 2-5. The red curve shows the prediction error with respect to ground truth data, with zoomed insets depicting actual predictions (blue) versus ground truth values (orange) to highlight the neural network's strengths and weaknesses.

in which it was trained, than the vertical configuration. In the vertical configuration, the neural network generalizes well in the Y-axis but shows more error in predictions in the X and Z axes. The prediction error of the neural network seems to follow a normal distribution, as seen in Fig. 2-7. From Table 2-2, the mean value of prediction error is very close to 0 N, indicating that most errors are small. Also, the low standard deviation values are reflected in Fig. 2-7 by the concentration of the probability density curves' peaks near 0 N. Overall, the prediction is smooth and accurate to about 1 N for the horizontal configuration and 2 N for the vertical configuration. This is about 10-20% of the range of the exerted force (around 10 N).

Table 2-2. Summary of key metrics for offline evaluation results (N). MAE, RMSE, Mean and STDev are calculated on the prediction error.

Configuration	Horizontal		Vertical	
X-component	MAE:	0.12	MAE:	0.50
	RMSE:	0.26	RMSE:	1.20
	Mean:	-0.02	Mean:	-0.47
	STDev:	0.26	STDev:	1.11
	Min:	-5.92	Min:	-9.95
	Max:	4.26	Max:	1.01
Y-component	MAE:	0.11	MAE:	0.16
	RMSE:	0.22	RMSE:	0.28
	Mean:	-0.03	Mean:	0.07
	STDev:	0.22	STDev:	0.27
	Min:	-3.67	Min:	-3.44
	Max:	4.83	Max:	3.15
Z-component	MAE:	0.16	MAE:	0.52
	RMSE:	0.40	RMSE:	1.28
	Mean:	-0.07	Mean:	0.43
	STDev:	0.40	STDev:	1.21
	Min:	-9.00	Min:	-7.02
	Max:	1.31	Max:	2.67

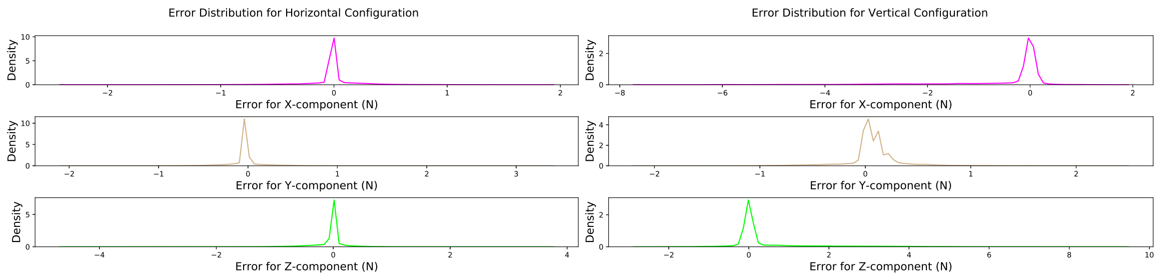


Figure 2-7. The probability density distribution of errors for both horizontal and vertical configurations. The density curve resembles a bell curve, revealing a normal distribution with the prediction error concentrating near 0 N for both horizontal and vertical configurations.

2.3.2.2 Haptic feedback pilot study

For online validation, we conduct a pilot study in horizontal configuration with three scenarios: (1) no force feedback, (2) force feedback from the neural network, and (3) force feedback from the force sensor under the phantom. One of the participants contributed to collecting the training data while the other two had not. The pilot study also tests the neural network's potential to generalize to novel users. The participants are asked to palpate three phantoms of low, medium and high stiffnesses, shown in Fig. 2-8. These phantoms are covered under a drape and their order is shuffled before a participant is engaged in a test scenario. Their heights are set to be equal, and decoys under the drape are used to prevent participants from guessing based on visual cues. For the no feedback case, the participants teleoperate with no haptic feedback upon contact. For the neural network and force sensor feedback cases, teleoperation occurs with force feedback from the neural network and the force sensor respectively. Force feedback is implemented by reversing the direction of the predicted force (either drawn from the neural network or the force sensor) and then applying it on the MTM.

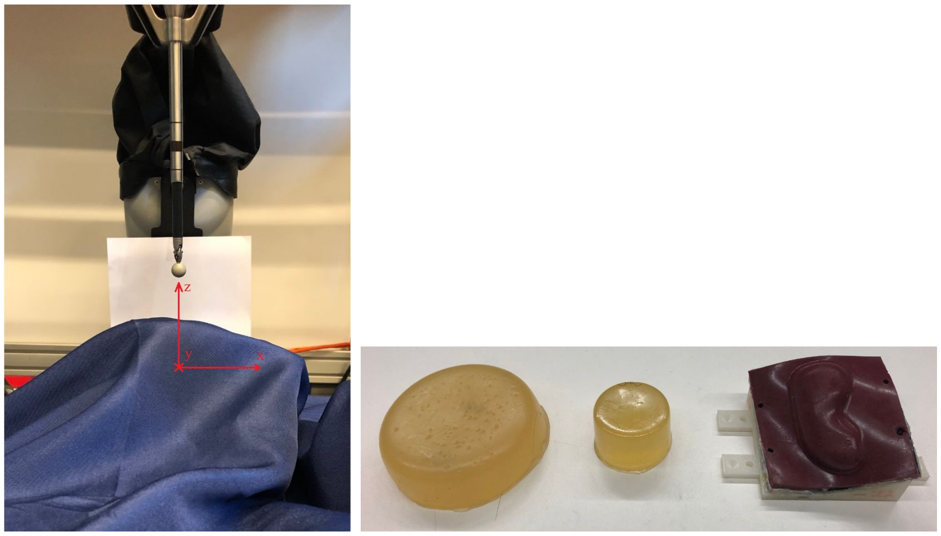


Figure 2-8. dVRK test bench horizontal configuration for pilot study. A small round marker is attached to the tip to prevent damage to soft phantoms. Also depicted are the phantoms used in the pilot study, with increasing order of stiffness from left to right. They are set to equal height and covered with a drape during the pilot study.

The participants are aware of the scenario they are currently in (no feedback, neural network feedback or force sensor feedback). In each scenario, while the test phantoms are presented to participants at random, participants are allowed to revisit previous phantoms as much as they need. Before proceeding to the next scenario, participants must give a final ranking of the three phantoms in the current scenario. A participant must be able to rank the stiffness of all 3 phantoms correctly and confidently in order to be given an A (able to discriminate). A participant who is able to do so with low confidence is given a G (correct but uncertain guess). A participant who is not able to rank all three phantoms correctly is given a U (unable to discriminate). Redoing a previously completed scenario is not allowed. The same process is applied for all three scenarios (no feedback, neural network feedback and force sensor feedback). Each participant takes part in the pilot study individually, with no other participants present for all three scenarios.

2.3.2.3 Results

This subsection demonstrates that the neural network's predictions, though prone to errors, are actually usable for palpating and stiffness discrimination in real-time conditions. Table 2-3 outlines the pilot study results, showing three scenarios from left to right respectively: no feedback, neural network feedback and force sensor feedback. Though allowed to revisit previous phantoms, the participants are able to finish all their scenarios in about 10 minutes (including the time to rearrange phantoms before each scenario). Participants of the user study are able to rank the stiffnesses of the three phantoms correctly when palpating using force feedback from the neural network and force sensor. With no force feedback, only one user, who had extensive experience with the dVRK, is able to discriminate correctly and confidently, while the remaining participants are either unable to rank or only able to guess correctly with high uncertainty. The neural network feedback does not feel as smooth as the feedback from the sensor, yet that does not affect the overall sense of touch and how the users rank the phantoms.

Table 2-3. Pilot study results. A: able to discriminate all three phantoms correctly and confidently, G: correct but uncertain in discrimination of all three phantoms, U: unable to discriminate all three phantoms

Participant	Test Scenarios		
	No Feedback	***NN Feedback***	***ATI Feedback***
I	U	A	A
II	G	A	A
III	A	A	A

2.3.3 Discussion

Our results demonstrate that our neural network is capable of providing useful force feedback without sensors at test time. We choose not to use joint positions in order to avoid the risk that the neural network learns a palpation map that is specific to the training phantom. Unfortunately, the results still show lower error in tests in the training setup vs. a novel one. More consideration is needed to generalize to a workspace outside of the training setup.

Through the palpation study, we show that the network has the potential to generalize to new phantoms on which it has not been trained. Although the user with the most experience with the system was able to distinguish the phantoms based on visual feedback alone, the network's predictions help less experienced users feel the stiffness of the phantoms. Using the neural network feedback, users obtained improvements comparable to using the physical force sensor.

Since the neural network has to learn a diverse range of contact scenarios as shown in Fig. 2-2, it gains some generalizability, with sacrifices in precision. Nevertheless, since RAMIS operations typically involve a wide array of contact situations, it is more imperative that our neural network is able to handle them all within reason. We discovered during the haptic feedback pilot study that consistent magnitude errors do not pose a significant issue. Rather, it is the neural network's flexibility allowing feedback at various contact angles, as depicted in Fig. 2-2 and Fig. 2-8, that provides a realistic sense of touch for the participants of the pilot study.

Although our study only analyzes two configurations, it can be argued that the data generated from these two configurations are quite representative of all possible scenarios in our test bench. For example, another potential vertical configuration is the mirror of the current vertical configuration, in which the phantom is rotated 90 degrees anticlockwise from the horizontal configuration. In this case, the rotation matrix applies changes but the network prediction should perform similarly.

While our network cannot learn the complete dynamics model as the mass/inertia matrix, coriolis/centrifugal forces, and gravity depend on the joint positions, our network's success in predicting forces without position as input suggests that the effect of the da Vinci PSM dynamics is relatively small. Other factors that depend primarily on velocity, such as friction, may have greater influence. Elements such as friction, backlash and hysteresis can be challenging to model accurately, as they are dependent on the individual robot arm. By casting the problem of internal dynamics modeling as a deep learning problem, we are able to reduce the complexity and create a predictive system that is robust to variations. We use a force sensor mounted under the phantom to approximate the force at the instrument tip. There may be some discrepancy, for example, due to the deformation of the intervening phantom.

Although our study has demonstrated the feasibility of an end-to-end neural network for estimating external force, one question remaining is how to implement such a system in practice. In particular, it remains to be seen how well a neural network trained on one PSM and one instrument will work for a different PSM and/or a different instrument, even of the same type. Retraining the neural network in the field, for example, prior to or during a surgery, would be challenging if the training requires a force sensor placed in the environment. In the next sections, we explore a self-supervised method to avoid this problem and consider how to deploy it in the operating room.

2.4 Indirect estimate - inverse dynamics identification

An alternate way to measure force is to estimate the torque it takes to drive the robot in free space, and then use the difference between the measured and predicted torque to estimate the force in interaction with the environment [32]. This section presents a neural network approach to estimate the inverse dynamics of the da Vinci PSM. The neural network can be used to control the PSM, for example to implement a computed torque controller, but the focus here is to estimate the external torques/forces acting on the joints by subtracting the internal torques/forces (neural network outputs) from the measured torques. The method is implemented and tested on the dVRK. Our approach makes use of the fact that the fundamental relationship between external forces and joint currents/torques is well known and this can be exploited to obtain accurate external force estimates from joint torque and position measurements without the use of external sensors for ground truth.

In the proposed method, identification is performed with the operator in the loop, as the operator controls the PSM, and dynamic identification is performed without a need for an automated excitation trajectory. This helps reduce the discrepancy between the surgical workspace and the excitation trajectory. Furthermore, the method is flexible as it can also provide a basis for deep neural networks that can be trained with data from different operations/instruments/surgeons.

2.4.1 Method

The force estimation method proposed in this section is composed of two parts. First, we telemanipulate the dVRK in free motion and use the data to train neural networks to estimate the inverse dynamic model of the PSM. Once the inverse dynamics is obtained, the identified dynamic torques are subtracted from the joint torque measurements. Then, we use the robot Jacobian to estimate external forces exerted on the end effector of the robot.

2.4.2 Neural network-based dynamic identification of the dVRK PSM

The dynamic model of a dVRK PSM can be described by the joint space equation:

$$M(q)\ddot{q} + C(q, \dot{q}) + G(q) + T(\dot{q}) + \tau_{int} = \tau \tag{2.2}$$

where q, $\dot{q}$ and $\ddot{q}$ represent the joint position, velocity and acceleration vectors, M, C and G denote mass/inertia matrix, Coriolis and centrifugal force/torque and gravity vectors, T represents the friction force/torque vector, τ_{int} is the internal force/torque vector representing the uncertain internal forces in the robot, and τ denotes the actuator force/torque vector, respectively. Since the dVRK uses tendon-driven mechanisms for both motion and force transmission, it is difficult to identify the dynamic model accurately due to the uncertain system parameters, friction, elongation and elasticity. As a result, the dynamic terms in (2.2) are not exactly known, however a lumped model can be defined to provide the sum of these effects:

$$\hat{\tau}_{dyn} = H(q, \dot{q}, \ddot{q}) \tag{2.3}$$

where q, $\dot{q}$, and $\ddot{q}$ are the model inputs, H is the lumped internal dynamics model of the robot, and $\hat{\tau}_{dyn}$ is the inverse dynamics torque estimate.

The crux of the proposal in this section is to obtain H, and to identify the lumped robot dynamics, with a set of neural networks. A neural network can be used as a black box model to approximate the nonlinear relationship between robot joint states (position, velocity) and the joint torques without the need for an explicit robot model. Currently, we approximate this function without acceleration measurements, as these measurements can be quite noisy, however they can also be used if good measurements are available. A separate neural network is used for each joint, with a total of six neural networks for the combined manipulator (excluding the gripper axis). Each neural network (see Fig. 2-9) includes:

- One input layer with twelve neurons for the position and velocity of each joint
- One hidden layer with 100 neurons
- One output layer with one neuron representing the predicted joint torque

Here, each neural network has input measurements from all the robot joints, but the output is the torque/force estimate for the respective joint. Thus each joint's identification error is used to train the respective neural network, and this provides better performance than a single neural network with multiple outputs where different

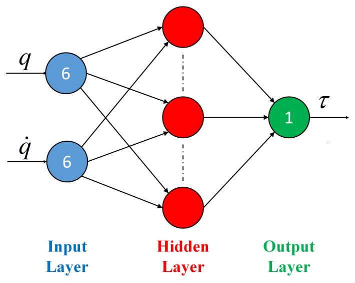


Figure 2-9. Neural network for inverse dynamics identification of one joint.

scales of the measurements become an issue. We do not consider a time window for this section and each set of inputs maps to the torque measured at the same time.

The learning process is achieved by back-propagation and Bayesian Regularization, as it can provide good generalization for difficult and noisy datasets [82]. In the multilayer network structure, we use the tan-sigmoid transfer function (tansig) and linear transfer function (purelin) for the hidden and output layers, respectively. Initial weight and bias values were selected randomly and then updated according to adaptive weight minimization (regularization) with the chosen algorithm.

The training dataset contains about 415,000 samples. In Fig. 2-10, the X and Y axes present velocity and position measurements, respectively, and the color scale shows the measured force/torque acting on the actuators. As the robot moves in free motion, the external force exerted on the robot is known to be zero, which means that the measured joint torques are purely due to inverse dynamics. The loss is the difference between the neural network estimate and the measured torques in free motion. The neural network learns to estimate the joint torque due to inverse dynamics in free motion. The optimal weights to minimize the error are found through back propagation, as shown in Fig. 2-11a. This scheme could be adapted for online training with adaptive neural networks [83].

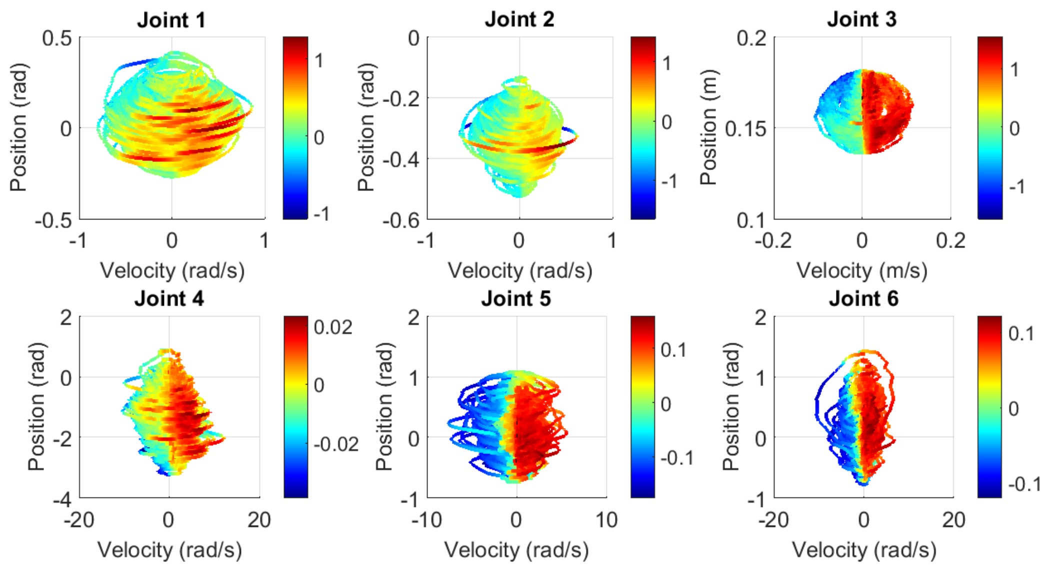
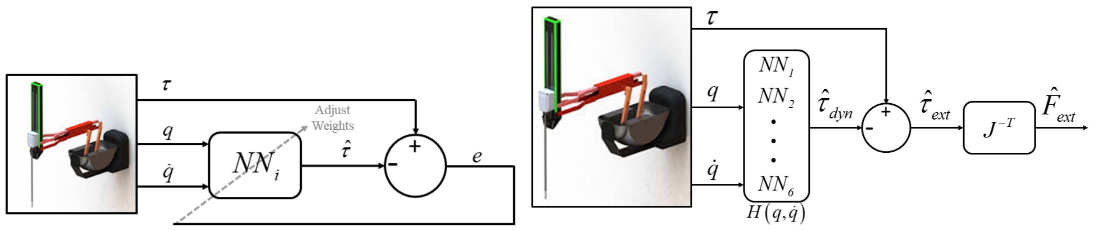


Figure 2-10. The relation between joint states (X and Y axes) and actuator force/torque (color scale) during training operation



(a) Offline training **(b)** External force estimation

Figure 2-11. (a) shows the offline training of each neural network and (b) shows how the networks are used for external force estimation after training.

2.4.3 External force estimation

When there is an external force/torque applied to the end effector of a surgical robot, the generalized dynamic equation in joint space is:

$$M(q)\ddot{q} + C(q,\dot{q}) + G(q) + T(\dot{q}) + \tau_{int} + \tau_{ext} = \tau \tag{2.4}$$

where τ_{ext} is the external force/torque vector acting on each joint. The external force/torque can be calculated by subtracting the inverse dynamics torque estimated by the trained neural network, $\hat{\tau}_{dyn}$ defined in (2.3), from the measured actuator force/torque τ:

$$\hat{\tau}_{ext} = \tau - \hat{\tau}_{dyn} \tag{2.5}$$

Using the Jacobian matrix of the robot (J) and the external joint torques/forces, the external force acting on the tool-tip in Cartesian space (see Fig. 2-11b) is:

$$\hat{F}_{ext} = J^{-T}\hat{\tau}_{ext} \tag{2.6}$$

We validate the trained network by comparing this estimate to force sensor measurements.

2.4.4 Experiments and results

This section describes the experiments conducted to validate the inverse dynamics identification and external force estimation. Also, a set of palpation experiments, performed on different phantom surfaces, for stiffness differentiation, are provided as a case study. To evaluate the performance of the estimation method, we use the normalized root mean square errors (NRMSE) between the actual and estimated forces/torques [70]:

$$NRMSE_i^{*} = \frac{\sqrt{\frac{1}{N}\sum_{n=1}^{N}[\hat{y}(n) - y(n)]_i^2}}{(y_{max} - y_{min})_i} \tag{2.7}$$

Here, N is the number of samples in the time series data from the experiments, y is the vector of reference force/torque, $\hat{y}$ is the vector of estimated force/torque. $y(n)$ and $\hat{y}(n)$ indicate the n^{th} samples of y and $\hat{y}$, respectively.

However, in [69] and [71], the identification performances were evaluated by calculating relative prediction error by the following formula:

$$RelE_i^{+} = \frac{\|y_i - \hat{y}_i\|_2}{\|y_i\|_2} = \sqrt{\frac{\sum_{n=1}^{N}[\hat{y}(n) - y(n)]_i^2}{\sum_{n=1}^{N}[y(n)]_i^2}} \tag{2.8}$$

Both formulas have been used here to make comparisons with the results in the mentioned papers. Errors with superscript $^{+}$ are computed using the relative prediction error (2.8), and those with the superscript * are calculated using the NRMSE (2.7).

2.4.5 Validation of dynamics identification

In the first experiment, the neural network outputs and the measured inverse dynamics forces/torques from the actuators in free motion are compared. For this experiment, a test data set was collected separately from the training set with the operator unilaterally controlling the robot in free motion for both datasets. It can be seen in Fig. 2-12 that the measured force/torque and estimated force/torque by the neural network are very close and dynamic identification is realized accurately on each joint with NRMSE of less than 10%, as shown in Table 2-4. Table 2-4 also shows the comparison of the errors between the proposed method (PM) and that of previous works. To serve as a reference, a single neural network (SNN), similar to [75], has been trained using the Levenberg-Marquardt method, with 12 inputs (joint variables) and 6 outputs (torques) and 100 hidden neurons. It can be seen that the results obtained with the proposed method are generally better than the prior results in the literature, and the use of neural networks for each joint provides an improvement over the SNN.

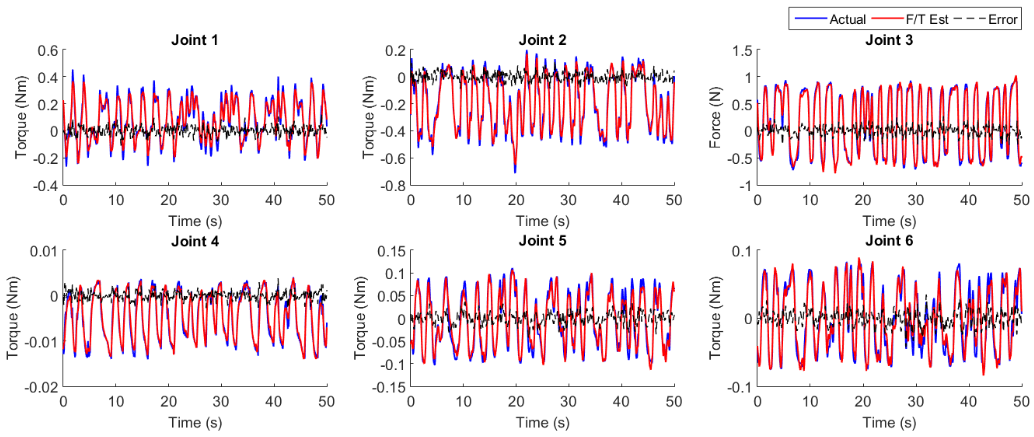


Figure 2-12. Joint torque identification results

2.4.6 Validation of external force estimation

The second experiment was conducted to validate the force estimation in the Cartesian X, Y and Z axes using a force sensor. Instead of phantoms from the above section, we

Table 2-4. Error values (* indicates NRMSE (Eq. 2.7) and $^+$ indicates RelE (Eq. 2.8)) in percentage of the measured range of joint force (f) or torque (τ) in free motion. Results are shown for the proposed method (PM) compared to using a single neural network (SNN) and other reported methods.

Method	τ_1	τ_2	f_3	τ_4	τ_5	τ_6
PM^*	4.37	3.51	4.99	4.70	6.28	6.80
[70]*	5.92	5.78	18.84	10.41	16.84	22.96
SNN^*	13.88	10.87	14.64	13.57	19.92	20.50
PM^+	18.40	11.19	13.95	12.10	21.23	22.39
[69]$^+$	22.07	31.55	29.55	11.93	35.10	45.30
[71]$^+$	9.30	17.80	19.10	13.40	23.90	21.30

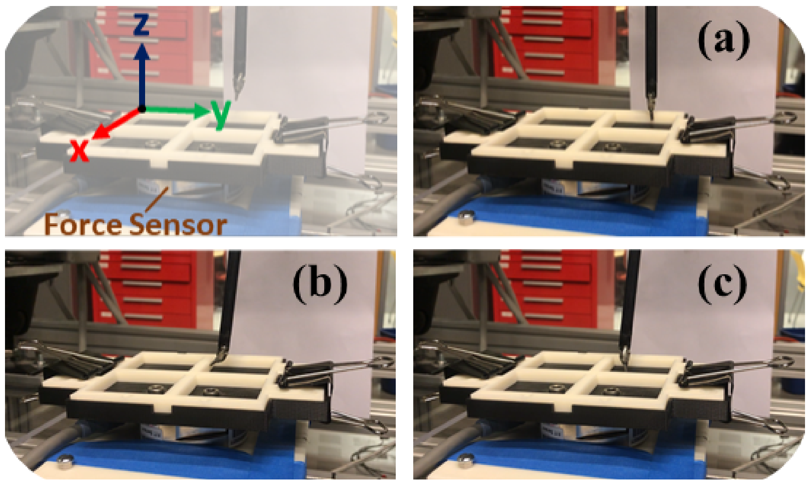
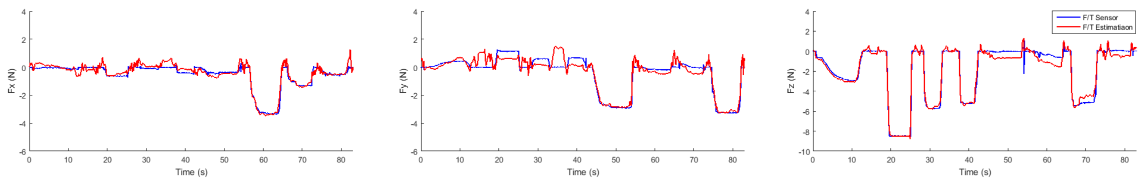


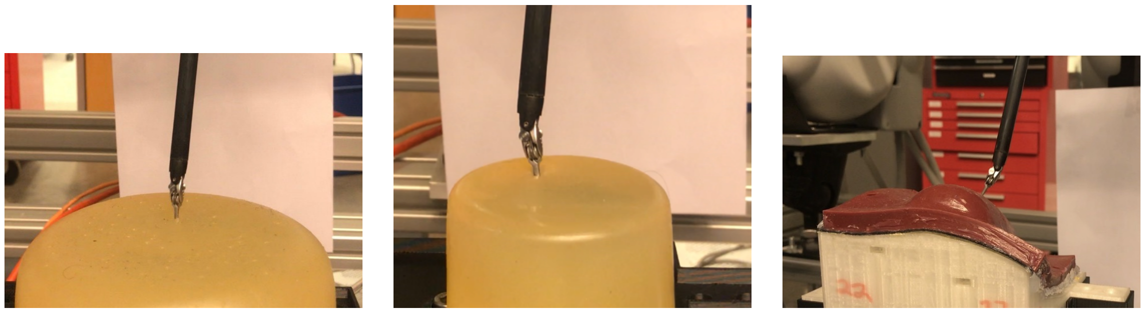
Figure 2-13. Test setup used in validation experiments. Contact in (a) X-axis, (b) Y-axis, (c) Z-axis

mount a rigid 3D printed apparatus with square holes to the sensor for the purpose of touching in each axis separately, as illustrated in Fig. 2-13. Figure 2-14 shows the comparison of measured forces and estimated forces when the end effector is in contact with the sensor, with results summarized in Table 2-5. When there is contact with the sensor, it can be observed that estimation results of each axis are in agreement with the force sensor outputs with less than 10% error in each axis. However, the fact that the robot was not always in direct contact with the sensor, but rather with an apparatus that was mounted on top of the sensor, and had contact at locations other than the tip, may account for some of the errors observed. Table 2-5 also compares



(a) Cartesian X-axis Force (b) Y-axis Force (c) Z-axis Force

Figure 2-14. Validation of external force estimation



(a) (b) (c)

Figure 2-15. Experiment setups used in stiffness determination: (a) Phantom #1, (b) Phantom #2, (c) Phantom #3

the force estimation errors obtained in this experiment with the results provided in [70]. While the exact experiment setup cannot be replicated, it can be seen that the performance of the proposed approach is comparable.

2.4.7 Stiffness identification

To demonstrate the feasibility of the proposed method for clinical purposes, we again palpate the phantoms used in the previous user study. To obtain a more objective measure, we calculate the relative stiffness values of three different phantoms by touching seven random points on each phantom (P1-P7), as shown in Fig. 2-15. The

Table 2-5. Normalized RMS error values (NRMSE) of Cartesian force (F) in contact and free-motion

Method	F_x	F_y	F_z
PM^*	6.80	9.86	4.45
[70]*	8.26	5.96	6.10

phantom in Fig. 2-15(a) has the lowest stiffness and the phantom in Fig. 2-15(c) has the highest stiffness. During the experiments, force and position data were recorded, external forces were estimated with the proposed algorithm and these were plotted with respect to changes in tool position, as shown in Fig. 2-16. Stiffness can be determined from the slope of these plots. As the exact stiffness values of the phantoms were not available, the same calculation was performed with the measurements from the force sensor placed under the phantoms. Table 2-6 shows consistent estimates of the average stiffness for each phantom by the proposed method (PM) and force sensor (FT).

This result shows that the proposed method can be useful in applications such as tissue differentiation which could be of practical use to surgeons.

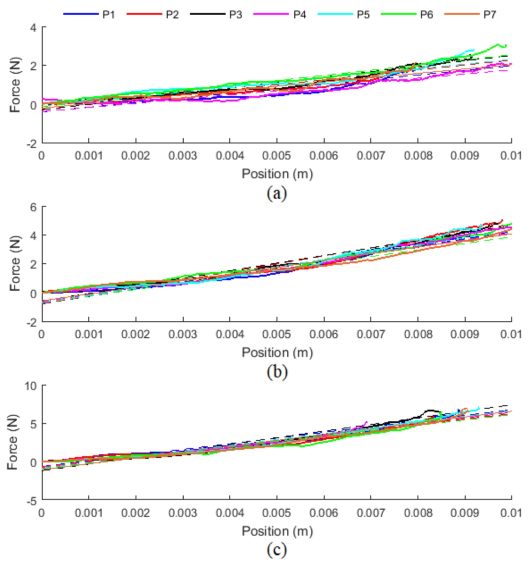


Figure 2-16. Stiffness differentiation results of: (a) Phantom #1, (b) Phantom #2, (c) Phantom #3

2.4.8 Discussion

When compared with existing identification and estimation results in the literature on the dVRK, the method in its current form has comparable or better error rates. However, the main advantage of the proposal is the learning nature of the identification

Table 2-6. Estimated stiffness values for each phantom using proposed method (PM) and force/torque sensor (FT), as well as the absolute difference between them.

Point	Phantom #1			Phantom #2			Phantom #3		
	PM (N/m)	**FT (N/m)**	**Abs Diff**	**PM (N/m)**	**FT (N/m)**	**Abs Diff**	**PM (N/m)**	**FT (N/m)**	**Abs Diff**
P1	237.7	238.8	1.1	499.2	388.6	110.6	738.1	799.1	61.0
P2	255.2	225.1	30.1	556.9	530.4	16.5	690.4	818.2	127.8
P3	281.5	250.0	31.5	525.6	517.5	8.1	857.7	806.8	50.9
P4	216.1	260.3	44.2	456.7	489.3	32.4	714.2	857.0	142.8
P5	243.8	252.2	8.4	479.4	533.2	53.8	759.9	806.0	46.1
P6	268.5	219.1	49.4	499.2	501.4	2.2	697.9	856.9	159.0
P7	224.0	245.3	21.3	465.5	522.7	57.2	761.5	807.7	56.2
Avg	**246.7**	**241.6**	**26.6**	**497.5**	**497.6**	**40.2**	**745.7**	**821.7**	**92.0**
Std	**23.4**	**14.9**	**17.7**	**35.0**	**51.7**	**37.7**	**56.5**	**24.7**	**49.0**

method and it can be improved with more datasets and training. It is also versatile when compared with other identification/estimation methods, such as the direct estimation method which requires a force sensor in the environment.

Since this method is self-supervised, the network could be trained online, even in the clinic. This would mean that training can be performed by the surgeon/operator as the method does not require an optimal excitation trajectory, given that sufficient excitation is provided by the human operator. The presented experiments were performed on a single PSM with a single instrument, so it is possible that real-time training updates may be required when applying the neural network to different PSMs and instruments. Also, during a surgical operation, robot dynamics is subject to changes as the robot is coupled with the environment at various contact points, including the trocar. In the next section, we further consider the implications of these coupling points.

Another limitation of the current implementation is that the training workspace did not cover the whole robot workspace, but this can be corrected by combining data from multiple training sets. Furthermore, this method can be augmented by gathering data from different dVRK and da Vinci setups around the world and training on the

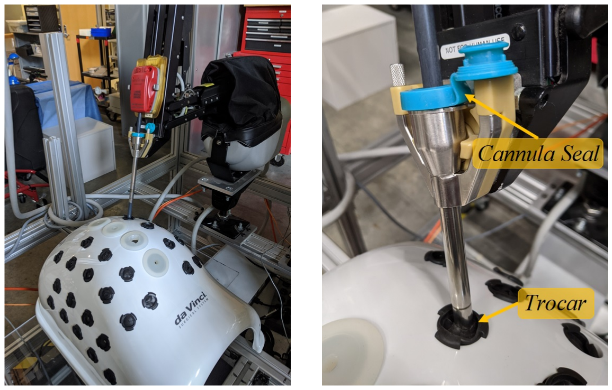


Figure 2-17. Left: Abdominal phantom with an instrument inserted through the trocar; Right: Closeup view of the trocar and cannula seal.

entire set. Finally, we have provided force estimates from the tip on three axes, but we explore extending the method to Cartesian torques in the next section.

2.5 Trocar force estimation

The method proposed in Section 2.4 estimated forces in free space and may not accurately account for trocar interactions in patients. While one could train the networks with surgery-specific data that includes trocar interactions, the time to collect sufficient training data and train the network may be clinically infeasible. In this section, we extend our use of neural networks to identify and eliminate trocar interaction forces [33]. As the trocar interaction forces are surgery-specific, we use two-step deep learning techniques to make this identification feasible in a clinical scenario. The proposed technique is self-supervised to not require additional setup in the clinic. In addition, we extend our prior work to estimate Cartesian torques in addition to forces. We test the proposed method on the dVRK system with an abdominal phantom.

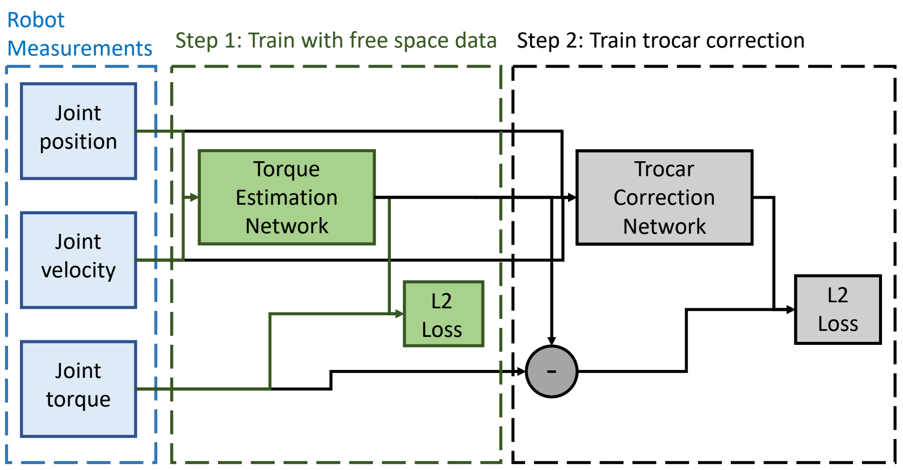

Robot Measurements
Step 1: Train with free space data
Step 2: Train trocar correction
Joint position
Joint velocity
Joint torque
Torque Estimation Network
L2 Loss
Trocar Correction Network
L2 Loss
-

Figure 2-18. Block diagram of two-step correction network. Step 1 estimates the torque for robot motion in free space. Step 2 adds a correction factor for the torque to account for the cannula seal and trocar interaction. It is trained from motion inside the trocar and learns the difference between the free space estimation and the measured torque.

2.5.1 Methods

To account for trocar interaction forces, we propose a two-step learning scheme. Fig. 2-18 shows the proposed setup. The first network, described in Section 2.5.1.1, follows from our indirect method for estimating free space dynamics described in Section 2.4.3. It is trained extensively for robot joint torque identification in the full robot workspace. It is followed by another network, described in Section 2.5.1.2, that learns patient and setup specific effects. The second network is designed to be trained with a subset of the workspace that is relevant to the procedure after the robot has been docked and the instruments have been placed through the port.

2.5.1.1 Step 1: joint torque identification network

The goal of the step 1 network is to learn the joint torque during free space motion. Following the method presented in Section 2.4, we train one network for each joint. We change the network for torque identification to LSTMs following the comparison in [84]. The input to the network consists of a sequence of positions and velocities of all the joints. The network architecture, shown in Fig. 2-19, consists of an LSTM layer with 128 hidden dimensions followed by two fully-connected layers with RELU activation.

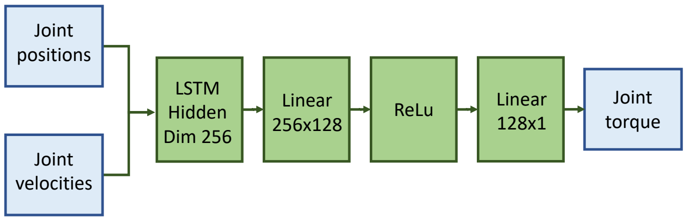


Figure 2-19. Free space network architecture. The input consists of a sequence of the positions and velocities for all joints. This goes to an LSTM layer, followed by two linear layers.

2.5.1.2 Step 2: compensation for trocar interaction forces

We insert the da Vinci Patient Side Manipulator (PSM) instrument inside an abdominal phantom such that its remote center of motion is at the insertion port. We also add a cannula seal to hold the instrument in the middle of the cannula. The setup is shown in Fig. 2-20. Since the trocar and the cannula seal introduce additional points of contact with the external environment, the torque identification learned on free space data may no longer be accurate. To correct for this additional interaction, we train another network for each joint based on the residual error between the measured torque and the torque predicted by the free space network, as shown in Fig. 2-18.

The second network has competing goals of making accurate predictions and learning from a small training set. While neural networks typically require a lot of data to train, the trocar interactions are unique for each surgery and can only be collected after surgical instruments have been inserted into a patient. We have to limit the surgery-specific training data set to data on the order of minutes, so that it is feasible to collect in the operating room. The training time must also be suitably short to fit into the clinical workflow.

Since LSTMs are good at capturing time series data but are generally slow to train and require more data, we use a feed-forward network for learning the surgery-specific trocar correction. To capture some time-series information without the heavy computation cost of LSTMs, we use a window of previous position and velocity measurements of all joints and the predicted free space torques as input. The network has two linear layers, with hidden dimension of 256 nodes. The small capacity is sufficient to learn the limited workspace for a particular patient setup and avoids overfitting to small training sets. We use ReLU activation between the hidden and output layers. The output of the network is added to the free space network prediction as a correction factor.

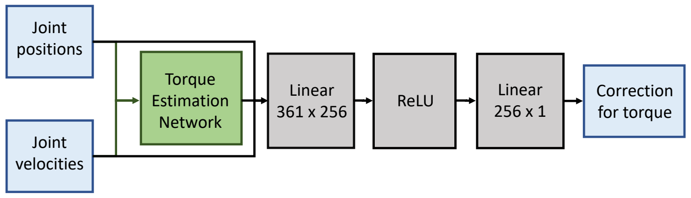


Figure 2-20. Trocar correction network architecture. The input consists of a sequence of the positions and velocities for every joint, and the output of the free space torque estimation network (Fig. 2-19) for the specific joint. This passes through two linear layers.

Once we train the network to estimate joint torques in the trocar, we follow the method outlined in Section 2.4.3 to estimate the Cartesian forces and torques.

2.5.2 Experimental setup

2.5.2.1 Free space torque identification network training

Extending our previous work, which only explored a limited workspace, we collected as full a workspace as possible and the workspace analysis is shown in Fig. 2-21. We record the data at 1 kHz using Rosbag and down-sample the data to 200 Hz. We implement the networks in Pytorch [85] and use the Adam optimizer with initial learning rate set to 0.001. We train the networks for 1000 epochs with a scheduler to decay on plateau.

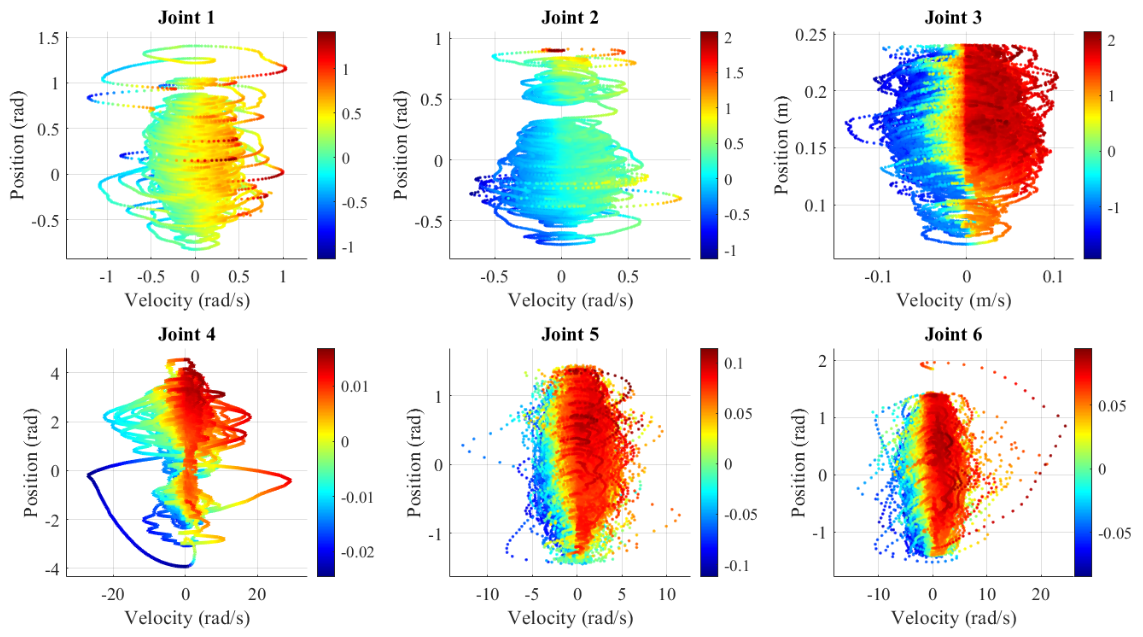


Figure 2-21. Joint states and torques used in training for free-space.

2.5.2.2 Trocar and cannula seal

To learn the trocar effects, we collect about 20 min of interactions inside the cannula and split it as 80%/10%/10% into train, validation, and test sets. We insert the cannula into a trocar hole in the abdominal phantom and place a cannula seal on top. Then, we attach the cannula to the dVRK and insert the instrument through the cannula. The abdominal phantom limits the workspace of the robot. The workspace inside the phantom is shown in Fig. 2-22. We observe that the ranges of the joint positions, particularly in joints 1 and 2, are limited by the phantom.

Since the free space network trained with the cannula seal matches the test conditions better, we expect that it will perform better than the free space network trained without the cannula seal. However, correcting the latter free space network enables us to estimate how well the correction schemes can generalize to test setups that are extremely different from training. The correction network is trained for 400 epochs with an initial learning rate of 0.0001. The window size was set to be 30 empirically through grid search. At 200 Hz, this gives 0.15 s of context for each torque

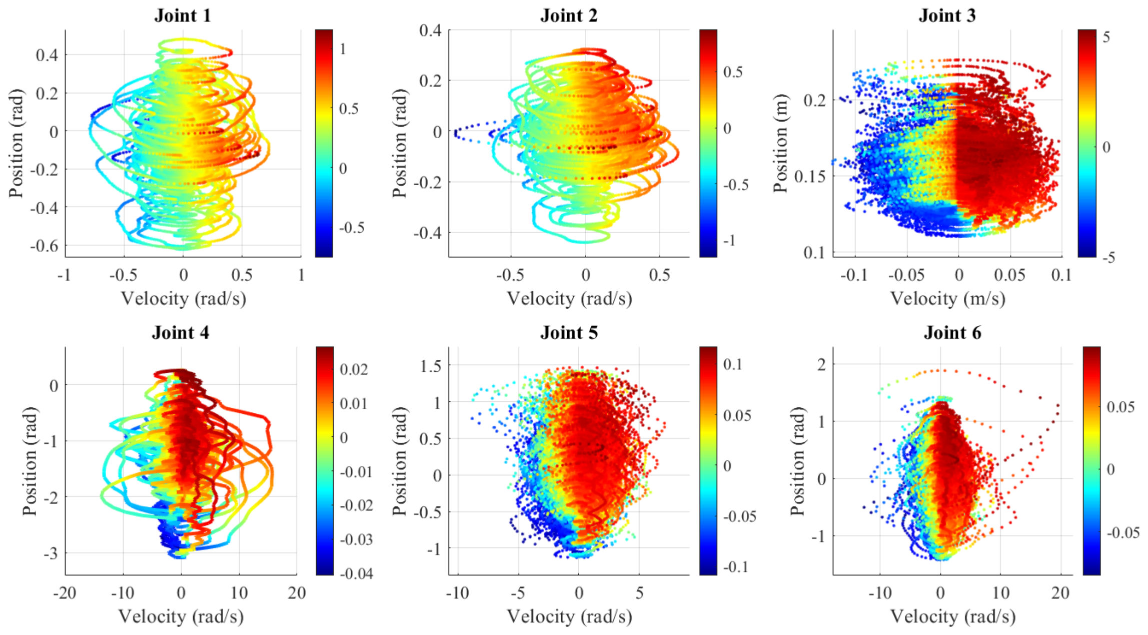


Figure 2-22. Joint states and torques used in training for inside the trocar. Note that the trocar workspace is confined by the phantom.

prediction. For comparison, we also train an LSTM on patient specific data (**Troc**), even though this may not be clinically viable due to the longer data collection and training times. These cases are described in Table 2-7.

Table 2-7. Tested network configurations

Troc	LSTM trained directly from trocar data using the network described in 2.5.1.1.
Base	LSTM trained on data collected without the trocar and without the cannula seal as described in 2.5.1.1
Seal	LSTM trained on data collected without the trocar but with a cannula seal using the network described in 2.5.1.1
Corr	Correction net trained from trocar data using second network as described in 2.5.1.2

2.5.2.3 Six degrees of freedom force estimation

We place a Gamma force/torque sensor in the workspace of the robot and mount a 3D printed structure on top as shown in Fig. 2-23. This structure enables us to use the da Vinci instrument to grasp a handle and apply both forces and torques to evaluate six degrees of freedom.

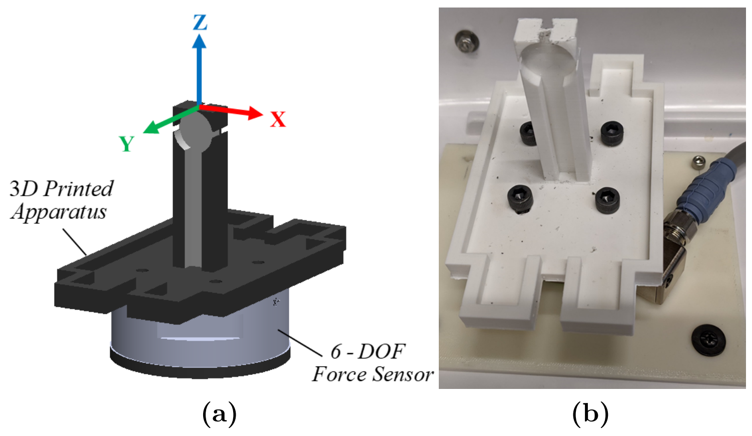


Figure 2-23. CAD model (a) and test setup (b) used to collect six degrees of freedom forces and torques

The coordinate systems of the force sensor and the robot are aligned, with just an offset in the Z direction. Thus, we can use a simplified adjoint matrix to compare the Cartesian forces and torques estimated at the robot instrument (F_{ri}) vs. the sensor (F_{fs}). The forces are the same and the Cartesian torques (moments) are given by

$$M_{fs} = M_{ri} + P \times F_{ri} \tag{2.9}$$

where P is the vector offset from the force sensor origin to the robot instrument origin.

2.5.3 Results

2.5.3.1 Training time vs. training dataset length

First, we analyze the runtime required to train the LSTM vs. the correction network given a set of training data of a certain length. The training was done on a Titan V GPU (Nvidia, Santa Clara CA, USA). Since the training can be parallelized, we train the networks for each joint in parallel and report the longest of the run times. We exclude the time to read the data from disk as we assume that the small amount of data can be collected and stored in memory when actually deployed.

For training data length of up to 4 min, the LSTM can be trained within 4 min and the feed-forward network can be trained within 40 s. For larger training sets, the LSTM exceeds the GPU memory and we reduce the batch size to 32. This increased the training time to up to 14 min for 18 min of data, indicating that the training data collection time is the main bottleneck for adapting to patient setups. With more GPU or larger GPU memory, the training could be further parallelized and the training time could become negligible. Since training the feed-forward network requires less memory use, the correction network can be trained within 2 min for up to 18 min of training data.

2.5.3.2 No-contact case

One of the key problems with not accounting for trocar forces is that they contribute falsely to forces in no-contact motion, so we consider this case first. Table 2-8 shows the results from no contact experiments performed in the trocar. The networks are trained with 10 min of data. In this condition, we see that the **Troc** case achieves the best performance. This could be because the LSTM learns smoother behavior than the feed-forward network and is less sensitive to the friction forces as the robot moves around with no contact. The correction network does improve upon the free space estimation, mainly in the F_x and F_y, which are most affected by the body wall (trocar). The correction network improves error in the insertion axis more in the **Seal** case than in the **Base** case. This is interesting as the cannula seal should have the greatest effect in that axis.

2.5.3.3 Force estimation

Lastly, we test contact force estimation inside the trocar by interacting with the force sensor and comparing it to our network predictions. We collect several long

Table 2-8. RMSE mean and (standard deviation) of Cartesian force (f, in N) and torque (τ, in Nm) in no-contact motion over 40 s. As the instrument's end-effector is moving in free space, the force and torque should be 0 in all cases. The estimated forces and torques result from poor filtering of trocar and cannula seal interactions. We note that the Base and Seal methods, which have not been trained with trocar interaction data, have higher errors.

Method	Troc	Base	Seal	Base + Corr	Seal + Corr
F_x	0.480 (0.322)	1.629 (0.924)	1.632 (0.926)	0.884 (0.593)	0.740 (0.503)
F_y	0.463 (0.313)	1.383 (0.870)	1.417 (0.870)	0.964 (0.632)	1.128 (0.765)
F_z	1.367 (0.815)	2.887 (1.327)	2.886 (1.330)	2.327 (1.017)	1.767 (1.201)
τ_x	0.036 (0.023)	0.125 (0.079)	0.125 (0.079)	0.086 (0.058)	0.109 (0.079)
τ_y	0.046 (0.033)	0.145 (0.079)	0.143 (0.080)	0.076 (0.051)	0.063 (0.042)
τ_z	0.016 (0.012)	0.053 (0.034)	0.057 (0.037)	0.030 (0.022)	0.032 (0.025)

interactions and separate them into fifteen 15 s segments. We calculate the RMS error for each segment and report the mean and standard deviation of the error over all trials. Table 2-9 shows the results for the **Base** and **Seal** cases, which do not consider surgery-specific training. Table 2-10 shows the results for the cases that do consider trocar data for different lengths of the training dataset.

Table 2-9. RMSE mean and (standard deviation) of Cartesian force (F, in N) or torque (τ, in Nm) for the two free space networks without correction when interacting with the phantom through the trocar (15 trials for each network).

	Base	Seal
F_x	2.363 (1.020)	2.363 (1.026)
F_y	1.485 (0.476)	1.479 (0.476)
F_z	3.295 (0.674)	3.300 (0.674)
τ_x	0.113 (0.011)	0.114 (0.010)
τ_y	0.146 (0.072)	0.152 (0.069)
τ_z	0.078 (0.018)	0.080 (0.020)

It can be seen that the correction networks have lower mean than the other approaches for the shorter training data sets. With more data, the LSTM trained directly on patient data (**Troc**) is able to get more accurate torque estimations compared to the smaller feed-forward networks. Fig. 2-24 shows a sample of the network predictions compared to the sensor readings. The training dataset length for

Table 2-10. RMSE mean and (standard deviation) of Cartesian force (F, in N) or torque (τ, in Nm) for different lengths of trocar training data when interacting with the phantom through the trocar (15 trials for each network).

	Time	Troc	Base + Corr	Seal + Corr
F_x	**4 min**	1.366 (1.284)	1.232 (1.264)	1.166 (1.236)
	8 min	1.277 (1.323)	1.231 (1.307)	1.148 (1.284)
	12 min	1.196 (1.304)	1.223 (1.306)	1.134 (1.286)
	16 min	1.172 (1.294)	1.222 (1.302)	1.138 (1.280)
F_y	**4 min**	1.189 (0.598)	1.025 (0.670)	1.235 (0.630)
	8 min	1.007 (0.459)	0.924 (0.387)	0.977 (0.368)
	12 min	1.033 (0.391)	0.947 (0.414)	0.994 (0.402)
	16 min	0.907 (0.363)	0.940 (0.461)	1.024 (0.444)
F_z	**4 min**	3.311 (0.782)	2.924 (0.716)	2.510 (1.006)
	8 min	3.317 (0.844)	2.903 (0.741)	2.468 (1.037)
	12 min	2.454 (1.109)	2.875 (0.758)	2.473 (1.053)
	16 min	2.382 (1.126)	2.878 (0.752)	2.472 (1.058)
τ_x	**4 min**	0.091 (0.023)	0.088 (0.026)	0.095 (0.024)
	8 min	0.083 (0.020)	0.081 (0.023)	0.084 (0.021)
	12 min	0.082 (0.023)	0.081 (0.023)	0.084 (0.021)
	16 min	0.077 (0.022)	0.082 (0.022)	0.083 (0.019)
τ_y	**4 min**	0.109 (0.086)	0.106 (0.087)	0.102 (0.087)
	8 min	0.106 (0.088)	0.104 (0.089)	0.101 (0.089)
	12 min	0.103 (0.090)	0.104 (0.091)	0.101 (0.091)
	16 min	0.104 (0.091)	0.103 (0.089)	0.100 (0.089)
τ_z	**4 min**	0.059 (0.025)	0.058 (0.023)	0.061 (0.025)
	8 min	0.056 (0.025)	0.054 (0.023)	0.057 (0.026)
	12 min	0.056 (0.027)	0.054 (0.024)	0.057 (0.026)
	16 min	0.055 (0.027)	0.054 (0.023)	0.057 (0.026)

the plots in Fig. 2-24 is 10 min.

Lastly, we evaluate the RMSE over the length of the training dataset used to train each method. Fig. 2-25 shows the accuracy in force estimation to illustrate how the RMSE changes as a function of dataset length.

2.5.4 Discussion

In our experimental setup, we observed that mimicking the test setup as well as possible is important since the correction network learned better correction in the **Seal**

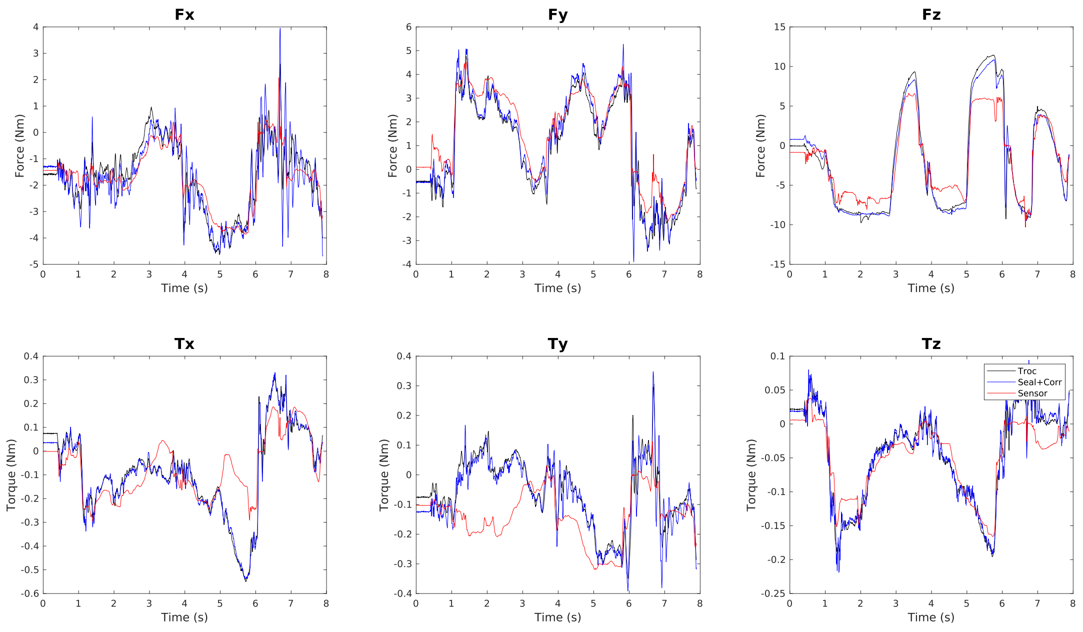


Figure 2-24. Cartesian forces and torques for directly-trained trocar network (**Troc**), the corrected torque when the free space network was trained with a cannula seal (**Seal+Corr**), and the force sensor measurements.

case compared to **Base**, even though both performed similarly without correction. We also observe that the improvement in Cartesian torques is limited. This may reflect a limited effect of the trocar on the wrist motions. The present experiments were performed with a single da Vinci instrument but future work aims to extend to more instrument types. We may see more differences in Cartesian torques between instruments. Since the free space network accounts for the robot dynamics, while the correction network should be mainly driven by the instrument shaft's interaction with the trocar and cannula seal, it is possible that the same correction network may work across different types of instruments.

Because the system can detect which instrument is installed, it is feasible to have a different free space network trained for each instrument type. There could even be different correction networks targeting different sources of error, such as to capture intra-type variations or in the same instrument over time. These changes could arise from different levels of wear and tear, and the cleaning and sterilization procedures.

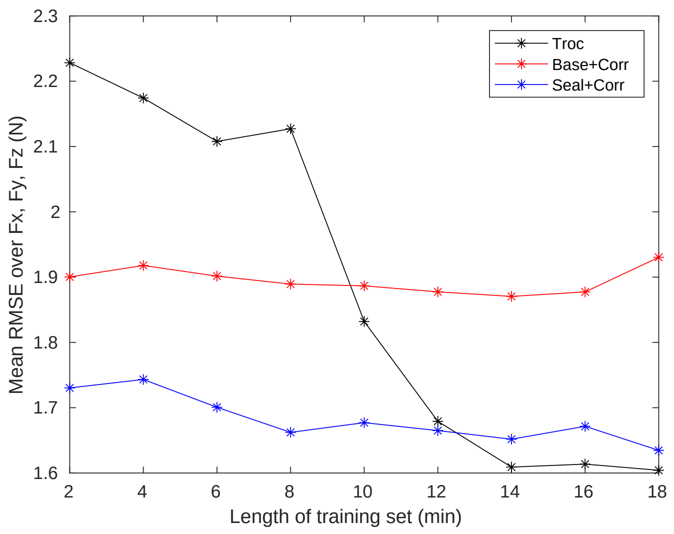


Figure 2-25. Training set length vs. mean error (RMSE) of estimated forces over Fx, Fy, and Fz.

While we have shown a correction network for the trocar, we can have a separate correction scheme to adapt to the instrument.

We observe from Fig. 2-25 that the error in **Troc** is higher than **Seal+Corr** with less than 12 min of training data, while the **Seal+Corr** has a low error throughout. This suggests that using a free space network with a correction step could reduce the amount of training data needed, making surgery-specific force estimation more clinically feasible. It is interesting to note that while the error of **Troc** reduces with more training data, as expected, the errors of **Seal+Corr** and **Base+Corr** do not show clear trends. This could be due to the estimates being noisier, as seen in the **Seal+Corr** case in Fig. 2-24. This noise may be due to the lack of memory between predictions since each torque estimate in the feed-forward network is made independently of previous estimates.

Unlike our phantom, patient body musculature and fat can show greater resistance and cause larger interaction forces at the trocar. Furthermore, the patient abdominal cavities are insufflated at the beginning of the operation and then the insufflation level changes during the procedure. These factors can cause changes in the interaction forces during the operation. Since the proposed method is self-supervised, it has the potential for lifelong learning to adjust for changes in insufflation levels. If the measured torques differ slightly from predicted, we can assume the motion is in free space and use it to update the network. Otherwise, if the measured torques differ greatly from the predicted, the change is more likely caused by contact. Computer vision techniques can be used as another check on contact vs. non-contact motion.

Another limitation of this work is that due to equipment limitation, we used the same cannula seal for training and testing. This gives an advantage to the **Seal** setup as the third joint is generally responsible for much of the applied force and is not affected by the trocar. However, we do demonstrate that in the absence of the cannula seal, our correction approach was able to improve the performance. This indicates that it could correct for changes in the cannula seal and account for more significant body wall forces in a patient as well.

hen used in surgery, the da Vinci Surgical System is in a busy operating room with more clutter than in our test setup. While we consider contact with the patient body, we have not tested for collision on the robot arm external to the patient. One method to filter these forces is to combine models of the surgical scene and the robot. With the patient model, we can build some expectation for stiffness in the anatomy. We could use the estimated force and displacement to estimate tissue stiffness at each point in time, such as done in Section 2.4.7. If there is a sudden force that creates a large jump in stiffness that falls outside the range we expect from the anatomy, we could reject the force measurement. If there is continuous excess force (such as from draping), we could use it to rebias our estimates.

Another requirement for using these methods clinically is to be robust to errors. While we experimented with directly reflecting the force feedback to the surgeon, qualitative feedback from the users indicated that it was jerky and detracted from the palpation task. Optimizing this force feedback in the closed-loop control system requires further study. An alternative method to represent forces is to visually display the force applied. We can use the force feedback to build a stiffness map of the organs, such as Chalasani et al. built using force sensors [86]. This would also allow the surgeon to reject errors in the measurement if they see a point where the stiffness is different from its neighbors. Using force estimation in conjunction with soft-tissue simulation methods, such as that presented in Chapter 4, could provide additional error checking. The measured force should match the expected force from the simulation.

2.6 Conclusion

In this chapter, we presented two methods for estimating forces in robot-assisted surgeries. We have demonstrated that neural networks can provide useful force estimation in both offline and online evaluation. The first method directly learns to predict forces from a sensor in the environment. It requires a sensor in the environment but does not require knowing the robot Jacobian. The second method is based on inverse dynamics identification for the da Vinci patient side manipulators. Using this identification method, we obtain a simple and robust way to filter out the dynamic components from the joint torque measurements for the estimation of the external forces. The method does not require a ground truth sensor and is easy to implement.

We also proposed a correction method for adjusting for surgery-specific interactions at the cannula seal and trocar. Clinical conditions, subject to additional external forces due to the cannula seal and trocar, can vary significantly based on the patient and clinical setup. With a few minutes or less of surgery-specific data, and 1 min training time, we estimate forces inside a phantom that are similar to those estimated with the direct method in Section 2.3. This suggests that the proposed method may be feasible to deploy in clinical use.

Future work will further evaluate the method's generalizability, such as testing on a different da Vinci arm and instruments of the same type to see whether different amounts of wear and tear would affect accuracy. Also, more tissue handling methods such as pulling, lifting and gripping the tissue can be explored, and may also require extension of the method to handle grip force estimation. It would be interesting to do online learning so that the network can adapt to changes in the patient condition over time. This approach could be the first step in creating a general force estimation method that can be adapted at test time to a variety of procedures.

Chapter 3

Interpreting surgeon actions

Robotic surgical systems such as the da Vinci Surgical System (Intuitive Surgical Inc., Sunnyvale, CA) are designed to improve surgical quality and patient outcome [87]. While these robotic systems improve certain aspects of surgery, such as the ergonomics, they are still acting as sophisticated extensions of the surgeon's arms. Specifically, these systems lack the ability to more actively support the surgeon. The predominant challenge that so far has prevented the adoption of intelligent assistance paradigms is the difficulty of contextualizing intraoperative information, such as laparoscopic video or robot kinematics data. A promising avenue towards contextualizing observations to provide active assistance is to recognize the surgeon's gestures and infer intent. This chapter looks at using the synchronicity of video and kinematics data streams. By using the two data streams readily available from robotic surgery as cross-modal, self-supervisory signals, we can learn a representation of the latent space of surgical actions without explicit gesture labels. The representation should reflect separations in surgeon gestures and skill levels.

Previous works in surgeon skill and gesture estimation have focused on task-specific, supervised learning, mostly either looking only at video or kinematics [88][89]. More recently, works have considered combining these two modalities [90] and adding system events [91]. Supervised-learning approaches have several drawbacks. A task-specific learning paradigm (such as supervised skill detection, for example) facilitates a narrow learning of the overall surgical process, possibly resulting in a narrow breadth of information retention beyond the specific training task. Further, supervised learning strictly depends on manual labeling, which limits the amount of data available, and more importantly, the potential to continuously learn throughout a procedure. Conversely, a self-supervised algorithm could benefit from lifelong learning and flexibly adapt to changes in surgical environments and surgeons. This representation leads to finding broader patterns in surgical action that may generalize across tasks.

Contributions

The contribution of this chapter is a self-supervised learning algorithm that learns task-agnostic representations of surgical gestures through cross-modal training from video to kinematics [92]. These representations capture information from the video and kinematics stream that is abstracted from the medium. We demonstrate that the learned representation captures semantically meaningful information by using it for gesture and skill recognition. While previous works have looked at estimating surgeon skill and gestures, they learn narrow characteristics that are not generalizable to different tasks. With our method, preliminary results suggest that the representations are meaningful even for gestures never seen in training. Since we can measure distances over the representations, our method can find similarities between new gestures and those that are in the training set. I wrote a skeleton code for this project while most of the implementation was carried out by Aniruddha Tamhane for his master thesis. I was a co-supervisor for his thesis project. After he graduated, I extended the results for the transfer learning section and refined the analysis and conclusions. The code for the methods presented in this Chapter can be found at `https://github.com/arcadelab/multimodal_learning`.

3.1 Related work

We review recent works in multi-modal self-supervised learning. To the extent of our knowledge, this is the first work to apply cross-modal learning to surgical robotics so we review related supervised and unsupervised learning approaches.

3.1.1 Multi-modal self-supervised learning

Cross-modal techniques learn representations by discovering transformations or correspondences between the same sequence of events recorded in multiple modalities. Examples include estimating video and optical flow to depth transformation, camera-motion to visual data relationship, and audio-visual correspondence [93]. In [94], Arandjelovic and Zisserman used an audio-visual correspondence task with triplet loss to train a two-stream, convolutional-based network to localize the area of the video generating the audio. Zhang and Lu discuss the difficulties associated with the triplet loss and propose a cross-modal projection matching loss instead [95]. Zhen et al. proposes learning modality-agnostic representations for retrieval tasks [96].

3.1.2 Supervised surgical skill and gesture recognition

A vast majority of research on surgical robotics task learning focuses on supervised learning. These methods learn a specific task, such as recognizing or forecasting gestures, from annotated data. In [88], Sarikaya and Jannin generate optical flow from surgical videos and use it to train a convolutional neural network (CNN)-based classifier for surgical actions. This shows that optical flow retains sufficient information of the scene while generalizing better to different backgrounds. In [97], Mazomenos et al. use Recurrent Neural Networks (RNNs) to classify surgical gestures. Similarly, in [89], DiPietro et al. applied unidirectional and bi-directional Long Short-Term Memory Networks (LSTMs) for gesture recognition. Long et al. demonstrate that higher accuracies could be achieved by combining video and kinematics using graph convolutional networks in [90]. In addition to considering these two signals, Qin et al. add system events [91]. In [98], a variety of traditional learning-based models are used to establish benchmarks for surgical gesture and skill recognition for the JIGSAWS dataset. In [99], a 3D CNN based architecture is used to classify surgeon skill.

3.1.3 Unsupervised surgical gesture recognition

In [27], DiPietro and Hager demonstrated that predicting the next surgical gesture using RNNs in an unsupervised manner captures the latent information necessary for surgical gesture recognition; however, uncertainty in action increases significantly after a short time. Further, they showed that these representations naturally cluster according to distinct higher level activities. Tanwani et al. use a Siamese network-based contrastive loss training objective to learn representations that group motions that are similar [100] and demonstrate a high efficacy in identifying suturing gestures. Murali et al. and van Amsterdam et al. both propose temporal and task-based segmentation of a sequence of video frames using deep-learning based image and kinematics features and Gaussian Mixture Model based clustering [101, 102]. These works demonstrate that using neural network-based features from both videos and kinematics works best for classifying gestures but it is unclear how each stream contributes.

3.2 Method

3.2.1 Problem formulation

We seek to learn a cross-modal representation of surgical robotic activity from video to kinematics data using self-supervised learning. Formally, given a dataset $\mathbb{D} = (\mathcal{V}_i, \mathcal{K}_i)_{i=1}^{i=n}$, consisting of tuples of surgical videos $\mathcal{V}$ and corresponding kinematics $\mathcal{K}$, we seek a lower-dimensional representation $r_i \in \mathcal{R}^d$ of surgical video $\mathcal{V}_i$. We hypothesize that this representation can be learned by using it to reconstruct the corresponding kinematics sequence $\mathcal{K}_i$. Reconstructing the kinematics sequence is not a useful task itself, as the kinematics information should be readily available in any robotic surgery. It is a proxy task for extracting the common information between the optical flow and the kinematic sequence, to produce a representation that is independent of both mediums. While this work has not explored using autoencoders, its goal is similar in that both aim to learn a dimensionality reduction that removes noise from the original representation.

We formulate the encoding as $r = \mathcal{E}(\mathcal{V}; \theta)$, where $\mathcal{E}$ is an encoder parameterized by some learned weights θ. The representation r should retain all the critical information pertaining to surgical manipulation, which we evaluate by decoding it, through decoder $\mathcal{D}$ parameterized by ϕ. We compare the decoding to the kinematics through a loss function $\mathcal{L} : \mathbb{R}^{|\mathcal{K}|} \times \mathbb{R}^{|\mathcal{K}|} \rightarrow \mathbb{R}$. Finally, we define our objective as the following loss minimization over θ and ϕ:

$$\min_{\theta,\phi} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(\mathcal{D}(\mathcal{E}(\mathcal{V}_i; \theta); \phi), \mathcal{K}_i) \, . \tag{3.1}$$

To learn representations that are independent of video-specific properties such as luminance, white balance and contrast, we choose to process optical flow rather than video sequences. To this end, we add a preprocessing step before the encoder to extract the optical flow $\mathcal{O}_i$ from the video frames $\mathcal{V}_i$ through transformation T. While images carry useful information in discerning objects and their boundaries, our method trains with the proxy task of reconstructing the kinematics. The image data does not add to this so we do not expect the network to make use of it. As such, we do not include it in our current input to the network though future work could consider a different proxy task that could make use of the image data. Using the $L2$ norm, denoted by $|| \cdot ||_2$, as the loss function $\mathcal{L}$ we rewrite Equation 3.1 as:

$$\min_{\theta,\phi} \frac{1}{n} \sum_{i=1}^{n} ||\mathcal{D}(\mathcal{E}(T(\mathcal{V}_i); \theta); \phi) - \mathcal{K}_i||_2^2 \, . \tag{3.2}$$

3.2.2 Dataset

We use the JIGSAWS dataset [103], which contains a series of annotated clips of surgical activity with corresponding kinematics sequences. The dataset is further sub-divided into three tasks (Knot-Tying, Needle-Passing and Suturing) performed using the da Vinci Surgical System. The dataset includes frame-level annotations for surgical gestures, anonymized surgeon identification and surgical skill metrics based on years of surgical experience. The kinematics are available on a frame-level and consist of 76-dimensional vectors that include the x, y, z coordinates of the left and right tool tips, the corresponding linear and angular velocities, the rotation matrix, and the gripper angle velocities. There are a total of 15 surgical gestures. Each task has a subset of the gestures. Knot-Tying, Suturing and Needle-Passing consist of 6, 10, and 8 gestures respectively. Although there was some class balancing in sampling gestures for the training set, the number of occurrences of each gesture varies. We refer readers to the dataset paper [103] for more detailed gesture descriptions and the exact split. Fig. 3-1 shows the complete list of gestures for each task. The surgeons have been classified into three levels of skill based on the amount of experience with robotic surgery as “novice”, “intermediate” and “expert”. The kinematics stream is synchronized to the video stream and both are reported at 30 Hz.

3.2.3 Network architecture

In previous work on cross-modal learning, the AVE-Net [94] parsed an image and audio snippet to predict whether the two were sampled from the same video. In initial experiments we found this approach to not perform well on surgical data, which we attribute to the observation that surgical gestures are often repetitive. Repetitive movements, such as suture placements, exhibit largely similar kinematics sequences so that unsupervised training via sequence matching as in AVE-Net is uninformative. Therefore, we transformed the training objective from an alignment to

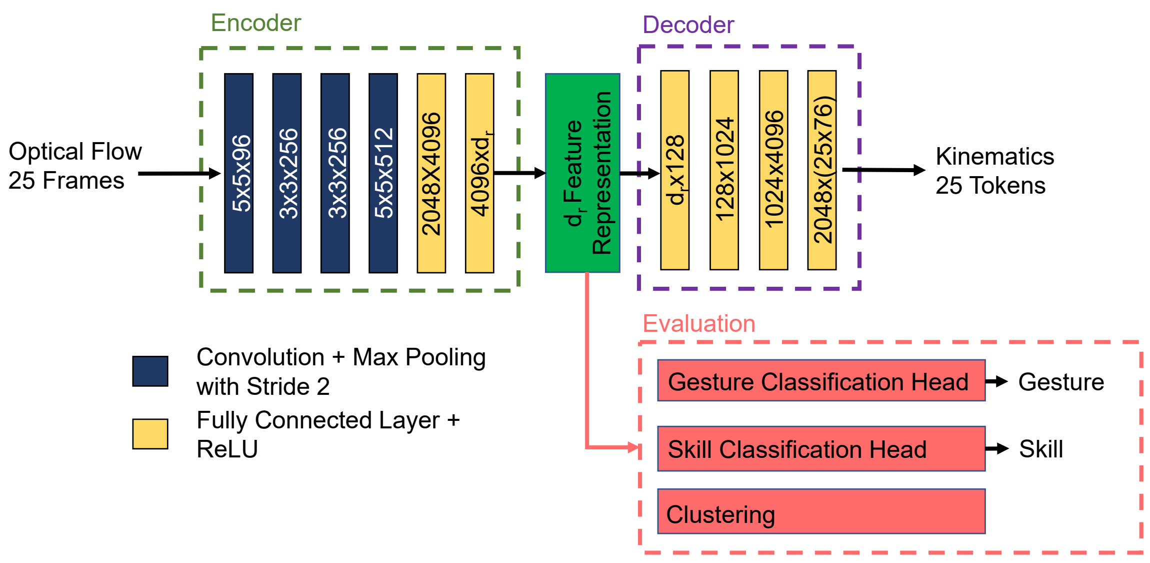


Figure 3-1. Cross-modal encoder-decoder network architecture. During training, the encoder and decoder are updated with unlabeled, but synchronized, video and kinematics frames. In testing, we freeze the encoder and decoder weights. Then, we investigate the structure of the feature representation as output by the encoder. This is done both qualitatively by visual inspection of latent representation using dimensionality reduction techniques and quantitatively by training classification heads for gesture and skill prediction.

a reconstruction task. We use an encoder-decoder architecture to match the video data to the kinematics through a bottleneck layer, from which we extract the desired high-level representations of surgical actions. A visual schematic of the architecture is shown in Fig. 3-1.

We preprocess the video by using the Farneback algorithm [104] to extract dense optical flow. Using optical flow instead of raw images can potentially filter out domain-specific information, such as the video quality and contrast, that should not affect representations of the surgical process but may inhibit generalizability. Fig. 3-2 shows an example of the pre-processing.

3.2.3.1 Encoder

We use a 2D CNN based encoder backbone with ReLU activation similar to the one in [105] to encode the optical flow. To capture temporal information, we stack

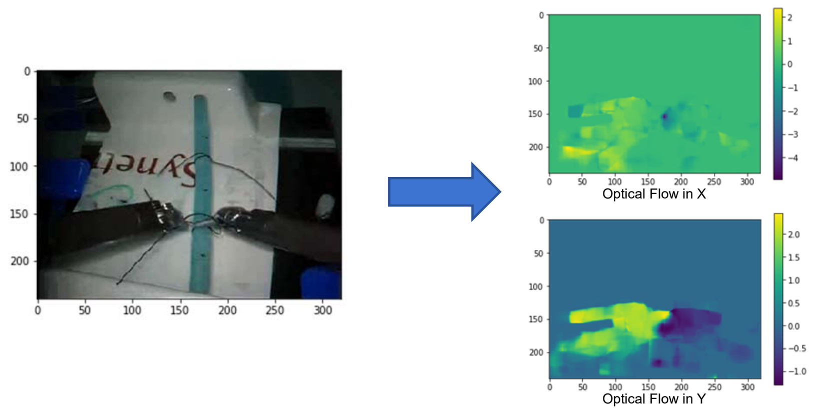


Figure 3-2. Pre-processing with optical flow from JIGSAWS dataset. This extracts a sparse signal and provides invariance to changes in background and illumination.

consecutive frames in the features channel. The encoder architecture is as follows: a 2D CNN layer with a filter of size 5×5 and stride 2 followed by a max-pooling layer with stride two, followed by two 2D CNN layers having filters of size 3×3, stride two and max pooling layers of stride 2. Finally, there is a 2D CNN layer with a 5×5 sized filter with a max-pooling layer of stride 2, followed by a fully-connected layer that yields a d_r dimensional representation vector.

3.2.3.2 Decoder

We choose a simple fully-connected network (FCN) with ReLU activations as the decoder. The decoder outputs the 25 kinematics vectors each corresponding to the sampled optical flows. We keep the decoder network relatively shallow to ensure maximum information retention in the representations yielded by the encoder network. The decoder is a 4-layer fully connected network with 128, 1024, 4096 and 1900 hidden layers. The ReLU activation function is used between each layer. The final hidden layer size of 1900 corresponds to the 25 76-dimensional gesture tokens that we seek to predict.

3.2.4 Evaluation tasks

In order to evaluate whether and to what extent the emerging representations are semantically meaningful, we consider transfer tasks relevant to robotic surgery. After unsupervised training of the encoder-decoder network, we freeze the weights and truncate the network after the representations layer. Then we examine the representation from the encoder layers of a test set through clustering and classification. First, we use dimensionality reduction techniques to plot the representations to visually inspect whether semantically meaningful patterns emerge. Then, we use the representations to train multi-class classifiers XGBoost, a tree-based gradient boosting classifier introduced in [106], to perform gesture and surgeon skill classification based on available ground truth annotation.

3.3 Experimental setup

3.3.1 Training

We first divide the frames in each video clip according to gestures. As gestures vary in length but our network requires a set number of frames as input, we extract the first 50 frames of every gesture. Gestures that were shorter than 50 frames were excluded. This excludes approximately 7% of the frames in each task. We then uniformly subsample from 50 frames by 2 to obtain 25 frame sequences, as a compromise between computational efficiency and information retention over a sufficiently long time span. We empirically determined these parameters through grid search. Given the frame frequency of 30 Hz in the original video, this approach encodes a gesture context of about 1.67 seconds, from which we extract the optical flow. We extract the 25 kinematics vectors corresponding to the video frames. The dimension of the learned representation r was set to 2048.

We use the Mean Squared Error (MSE) metric between the kinematics vectors and the decoder output as training loss, as given in Equation 3.2. We use the Adam optimizer with learning rate set to 10^{-3} and with a weight decay parameter (equivalent to an L2 weight penalty) of 10^{-8}. We train the networks for 1000 epochs and use a batch-size of 128 during our training to fit on an NVIDIA Titan Black GPU. Lastly, we follow the JIGSAWS proposed balanced leave-one-trial-out data split for train and test set for each task. In each split, one trial is left out for test data. To construct the training set, the dataset provides a list of random extracted gestures among the remaining trials to provide a consistent number of gestures in each class between the splits. We train and test over all proposed splits and average the results.

We train our encoder-decoder network on gestures from one task at a time as well as all the three tasks together. We found that there was a labeling error in the original Needle-Passing task in the JIGSAWS dataset. Instead of using the provided labels, we match each gesture sequence specified in the "Experimental Setup" section of the JIGSAWS dataset with those specified in the individual task sections. We look for a correspondence between the start frame, end frame, gesture label, surgeon label and trial labels in both the sections. While we found correspondence for all gestures sequences for Knot-Tying and Suturing, there was no correspondence for any of the Needle-Passing gestures. Therefore, we perform our own version of the leave-one-trial-out experiment. The scores reported for Needle-Passing (marked by an asterisk in the results) are an average of scores obtained by leaving each of the users from the training data and testing on a combination of the user's data left out of the training scheme and 20% of the rest of the data held out from the testing.

3.3.2 Evaluation

We test the quality of our learned representations for a number of tasks as a measure of the holistic representations of the surgical process. To obtain a visual representation of the information encoded in these representations, i. e., the output of the encoder, we embed them in 2D space using the U-MAP algorithm [107]. We choose U-MAP over alternatives because it preserves local as well as global distances in the lower dimensional latent-space, and enables processing of instances not used for the initial embedding. In the resulting embedding plots, ground truth annotations including skill and gestures are used to distinguish different surgically meaningful events. The emergence of clusters in this visualization would demonstrate that the representations learned during unsupervised learning discover information that is pertinent to surgical processes, as the labels are not used in the training process. We then demonstrate the usefulness of the learned representation by training a classifier that maps the representation to labeled tasks (gesture and skill classification). Since the goal of the classifier is to assign labels based on the clusters already existing in the representation, we limit the tuning of the classifier. We set the number of estimators to be 1000 and keep the default values for the remaining hyperparameters. We use the full 1024-feature representation as input to the classifier. Using U-MAP for dimensionality reduction is purely for visualization. Information is lost during the dimensionality reduction and using the resulting two-dimensional representation gives worse classification.

3.4 Results

3.4.1 Representations visualization using U-MAP

We plot the 2D projections of the representation from the Knot-Tying, Needle-Passing, and Suturing tasks, color-coded separately for different gestures and skill levels in Fig. 3-3. The gestures and corresponding colours are defined in Table. 3-1. We note that in each of the skill-based plots, Fig. 3-3(b,d,f), the representations neatly cluster into two distinct skill-based clusters corresponding to “novice” and “expert” skill level, with the representations from the “intermediate” skill level spread across both these clusters. A possible explanation for this phenomenon is that the skill “intermediate” is a vague category and encompasses people with varying skill-levels that may be closer

Label	Gestures	Knot-Tying	Needle-Passing	Suturing
G1	Reaching for needle with right hand	Blue	Blue	Blue
G2	Positioning needle	-	Orange	Orange
G3	Pushing needle through tissue	-	Green	Green
G4	Transferring needle from left to right	-	Maroon	Maroon
G5	Moving to center with needle in grip	-	Purple	Purple
G6	Pulling suture with left hand	-	Brown	Brown
G8	Orienting needle	-	Pink	Pink
G9	Using right hand to help tighten suture	-	-	Gray
G10	Loosening more suture	-	-	Olive
G11	Dropping suture at end and moving to end	Orange	Gray	Sky blue
G12	Reaching for needle with left hand	Green	-	-
G13	Making C loop around right hand	Maroon	-	-
G14	Reaching for suture with right hand	Purple	-	-
G15	Pulling suture with both hands	Brown	-	-

Table 3-1. Gestures in each class and the corresponding colours

to the "novice" or the "expert". This is in line with previous supervised methods that classified intermediate surgeons with the highest error rate [108]. Another interesting observation from the gesture-based plots, Fig. 3-3(a,c,e), is that each individual gesture (again distinguished by different colors) has two distinct clusters corresponding to a distinct skill category. Thus, this visualization strongly suggests that each gesture has a unique representation depending on whether it has been performed by an "expert" or a "novice".

3.4.2 Skill recognition

We train the XGBoost classifier on the representations for a 3-class classification task on the self-reported user skill. We freeze the encoder weights while training the classifier and repeat our experiment on the train-test leave-one-trial-out splits specified in the JIGSAWS dataset. We report our findings as an average of the results obtained on the splits in Table 3-2. We include comparison to results from a fully supervised algorithm, 3D-CNN (OF) [99] with the leave-one-super-trial-out result. The results

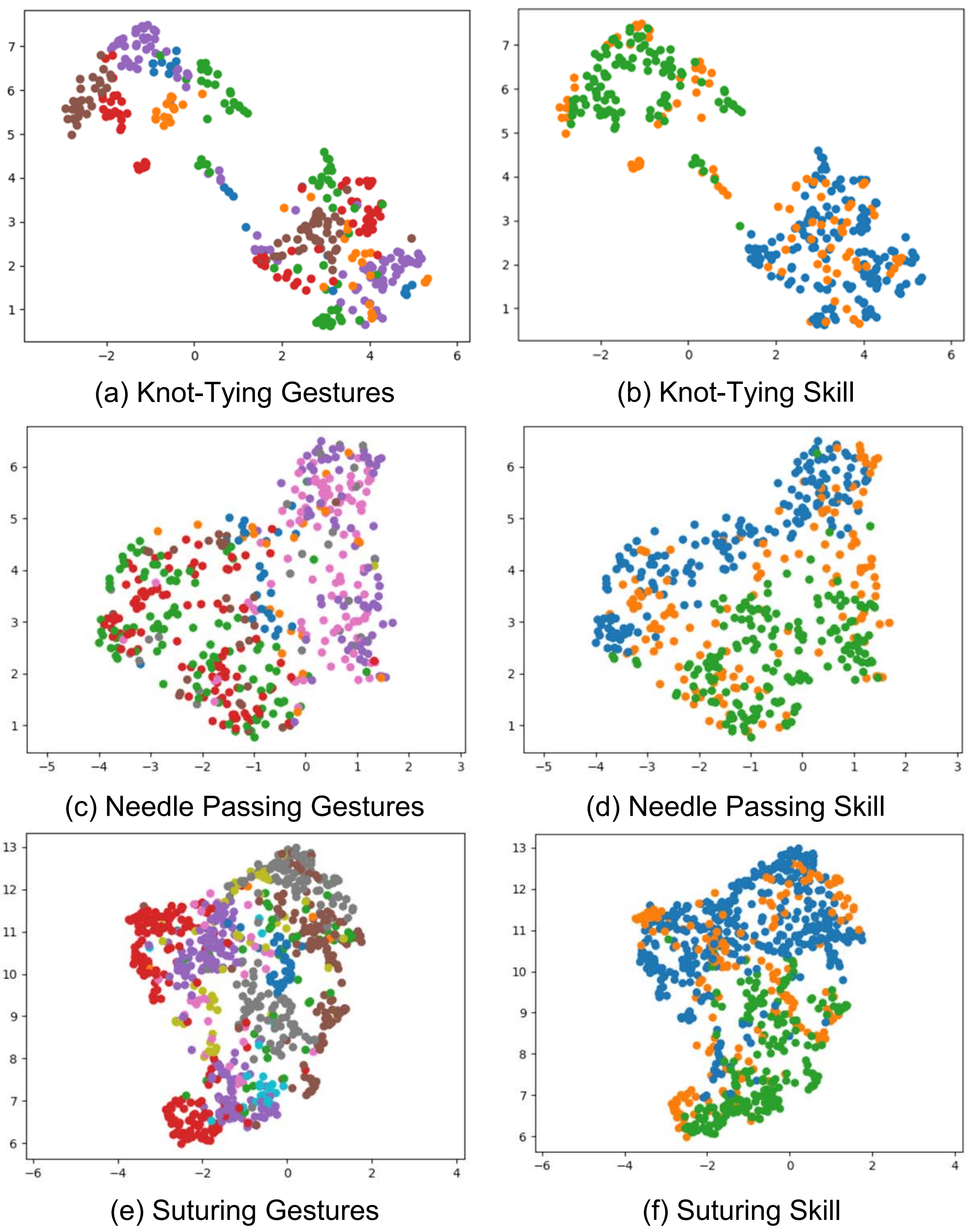

(a) Knot-Tying Gestures
(b) Knot-Tying Skill
(c) Needle Passing Gestures
(d) Needle Passing Skill
(e) Suturing Gestures
(f) Suturing Skill

Figure 3-3. 2D U-MAP plots of the representation vectors for all 3 JIGSAWS tasks. The skill based color coding is as follows: "novice" (blue), "intermediate" (orange) and "expert" (green). Colors in the left plot are assigned gestures specific to that task and are defined in Fig 3-1.

indicate a significant retention of information related to surgeon skill though it lags behind the results from fully-supervised approaches. As seen in Table. 3-3, a major portion of the classification error is contributed by surgeons having an "intermediate" skill. This reaffirms the observation made in Section 3.4.1 that the representations falling in this particular skill category follow a uniform spread from "novice" to "expert" and do not necessarily cluster tightly, unlike the other two skill levels. This makes it significantly harder to classify.

Table 3-2. Skill recognition results (mean $\pm$ standard deviation) on representations learned on each task as well as a combination of all the tasks. Since the JIGSAWS split is based on leave-one-trial-out, each test set contains videos of different actions but only one skill level. Supervised skill classification results (3D-CNN (OF)) [99] have also been reported for comparison. *We use a different split from JIGSAWS.

Test Dataset	Train Dataset	Accuracy	3D-CNN (OF)*
Knot-Tying	Knot-Tying	0.81 ± 0.05	0.95 ± 0.03
Suturing	Suturing	0.67 ± 0.03	1.00 ± 0.00
Needle-Passing*	Needle-Passing*	0.63 ± 0.08	1.00 ± 0.00
Knot-Tying	All	0.63 ± 0.06	NA
Suturing	All	0.66 ± 0.04	NA
Needle-Passing*	All	0.48 ± 0.06	NA

Table 3-3. Confusion matrix for skill levels for each gesture for each task: novice (N), intermediate (I), and expert (E)

		Estimated Label		
		N	I	E
True Label	N	20	4	2
	I	13	14	4
	E	18	6	12

(a) Knot-Tying

		Estimated Label		
		N	I	E
True Label	N	31	8	0
	I	9	20	7
	E	1	3	41

(b) Suturing

		Estimated Label		
		N	I	E
True Label	N	104	5	5
	I	12	20	10
	E	6	2	35

(c) Needle-Passing

3.4.3 Gesture recognition

Similar to the skill recognition task, we train an XGBoost classifier on the representations for gesture classification. We repeat our experiment on the train-test splits specified in the JIGSAWS dataset with a leave-one-trial-out splitting scheme. We report our findings as the aggregate of the results obtained on the splits in Table 3-4. For comparison, Table 3-5 shows the gesture-classification results from the fully supervised method, LDS (vid) [98], with the leave-one-super-trial-out split.

Table 3-4. Gesture recognition results on representations learned on the respective tasks as well as a combination of all the tasks. *We use a different split from JIGSAWS

Test Dataset	Train Dataset	Accuracy	Precision	Recall	F-1
Knot-Tying	Knot-Tying	0.64 ± 0.03	0.67 ± 0.06	0.64 ± 0.03	0.60 ± 0.04
Suturing	Suturing	0.68 ± 0.03	0.74 ± 0.02	0.68 ± 0.03	0.68 ± 0.03
Needle-Pass.*	Needle-Pass.*	0.64 ± 0.03	0.64 ± 0.04	0.64 ± 0.03	0.63 ± 0.03
Knot-Tying	All	0.70 ± 0.03	0.73 ± 0.03	0.70 ± 0.03	0.70 ± 0.03
Suturing	All	0.67 ± 0.03	0.74 ± 0.03	0.67 ± 0.03	0.68 ± 0.03
Needle-Pass.*	All	0.59 ± 0.04	0.58 ± 0.05	0.59 ± 0.04	0.57 ± 0.04

Tables 3-6, 3-7, and 3-8 show the confusion matrices for the true vs. estimated gesture labels for Knot-Tying, Suturing, and Needle-Passing respectively. We note that some gestures such as G5 and G10 in the Suturing task are particularly difficult to classify as they have very few examples. Other gestures such as G3, "pushing needle through tissue", tend to be well classified. This may be because optical flow captures the tissue motion, and many pixels contain movement. The reverse is true for G8, "orienting needle", which contains very sparse information in the optical flow input.

Table 3-5. Gesture recognition results using supervised learning (LDS (vid) model [98]) have been provided below for comparison.

Dataset	Accuracy	Precision $\pm$ std.
Knot-Tying	0.89	0.91 ± 0.07
Suturing	0.90	0.82 ± 0.30
Needle-Pass.	0.67	0.73 ± 0.15

3.4.4 Transfer learning

We measure the robustness of our learned algorithm through a transfer-learning based gesture recognition task. We generate representations of one dataset (e. g., Needle-

Table 3-6. Confusion matrix for true vs. estimated gestures in Knot-Tying

		Estimated Label					
		G1	G11	G12	G13	G14	G15
True Label	G1	2	0	1	0	4	0
	G11	0	9	1	0	0	0
	G12	0	2	11	2	2	0
	G13	0	0	1	14	1	2
	G14	3	0	4	0	17	1
	G15	0	0	3	1	0	12

Table 3-7. Confusion matrix for true vs. estimated gestures in Suturing

		Estimated Label									
		G1	G2	G3	G4	G5	G6	G8	G9	G10	G11
True Label	G1	9	0	0	0	4	0	0	0	0	0
	G2	0	39	2	0	1	0	1	0	0	0
	G3	0	1	38	0	0	0	0	0	0	0
	G4	0	0	0	18	0	2	1	0	0	0
	G5	1	2	2	0	4	0	2	0	0	0
	G6	0	0	2	3	0	37	1	0	0	0
	G8	0	2	1	3	0	2	3	0	0	0
	G9	0	3	0	0	0	1	0	2	0	0
	G10	0	1	0	0	0	1	0	0	0	0
	G11	0	1	0	7	0	2	1	0	0	3

Passing) using an encoder network trained on a different dataset (e. g., Knot-Tying). To show whether the encoder learns meaningful representations for gestures it has never seen before in training, we plot the new gestures on the U-MAP of the dataset used to train the encoder, shown in Fig. 3-4. The only gestures that the Knot-Tying and Needle-Passing tasks share are G1 and G11. These are introduced by the JIGSAWS dataset and used to begin and end trials. Interestingly, the new gestures are mostly in the space of "novice" users, which intuitively makes sense as there is more variation in gestures at that skill level. This suggests that the networks learn to attribute more certainty to expert gestures and could be a step in identifying new gestures for life-long learning. We also train skill and gesture classification heads on the representations of

Table 3-8. Confusion matrix for true vs. estimated gestures in Needle-Passing

		Estimated Label							
		G1	G2	G3	G4	G5	G6	G8	G11
True Label	G1	6	0	0	0	1	1	0	0
	G2	0	23	2	1	1	1	0	1
	G3	0	3	28	0	1	1	0	0
	G4	0	0	0	11	0	5	0	0
	G5	0	2	1	0	2	0	0	0
	G6	0	3	0	3	0	12	0	0
	G8	0	1	3	0	0	1	1	0
	G11	0	1	0	0	0	1	0	3

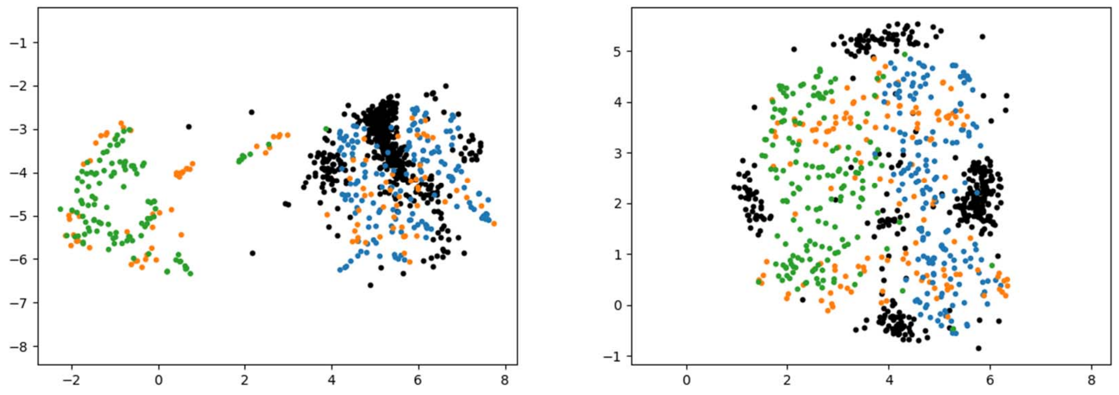


Figure 3-4. U-MAP plots of representations of Needle-Passing gestures encoded by a network trained solely on the Knot-Tying task and vice-versa. Gestures from the test task are shown in black. The train task representations are colored by surgeon skill: "novice" (blue), "intermediate" (orange) and "expert" (green). New gestures are mapped mainly to "novice" actions which inherently contain more variation than "expert" actions.

the unseen task. We report our findings as an aggregate of the results of the 5 random split based experiments in Table 3-9. We observe that the Needle-Passing and Suturing tasks have a large number of gestures in common (7 out of 10) while there are a few common gestures between either task and Knot-Tying. This may partially explain the higher accuracy reported for transfer learning of the representations learned on the Needle-Passing task that were tested on the Suturing task. It is important to note that the background and phantom changes between the tasks.

Table 3-9. Results for gesture classification on representations generated using transfer learning

Test Dataset	Train Dataset	Accuracy	Precision	Recall	F-1
Needle-Pass.	Knot-Tying	0.45 ± 0.05	0.46 ± 0.06	0.44 ± 0.05	0.43 ± 0.05
Suturing	Knot-Tying	0.62 ± 0.02	0.63 ± 0.05	0.62 ± 0.02	0.59 ± 0.02
Knot-Tying	Suturing	0.65 ± 0.04	0.65 ± 0.04	0.65 ± 0.04	0.65 ± 0.04
Needle-Pass.	Suturing	0.45 ± 0.05	0.46 ± 0.04	0.45 ± 0.05	0.43 ± 0.05
Knot-Tying	Needle-Pass.	0.57 ± 0.03	0.61 ± 0.05	0.57 ± 0.03	0.57 ± 0.04
Suturing	Needle-Pass.	0.60 ± 0.02	0.57 ± 0.03	0.60 ± 0.02	0.56 ± 0.03

3.4.5 Comparison to supervised methods

While developing this method, we were simultaneously in discussion with collaborators working on a fully-supervised gesture recognition for JIGSAWS [90] that uses the kinematics and video streams from both cameras as input. The method proposed relies on a multi-step network, with separate temporal convolution layers, LSTMs, and a message-passing graph as the last layer to learn contributions from the different modalities. This method obtains 88.1% and 87.9% accuracy on the Knot-Tying and suturing tasks respectively.

One drawback of the method is that although we aimed to learn across datasets collected at Johns Hopkins University and the Chinese University of Hong Kong simultaneously, we were not able to fully address the different setup and calibrations between the two systems. As such, we could only state that the same method could be trained on each data set separately and show similar performance on both. It is unclear how each dataset could benefit from learning from the other. On a single setup and task though, the fully-supervised method obtains better performance compared to the unsupervised one.

3.5 Discussion and future work

This work introduces an encoder-decoder based self-supervised learning paradigm that learns semantically meaningful representations of surgical actions from optical flow by reconstructing kinematics from video. We train the network on the JIGSAWS dataset and show that visualizing the resulting representations using U-MAP forms clusters based on surgeon skill and gestures, despite the fact that the network has never been trained on either label. While the gesture classification shows less distinct clustering than skill, we believe this may be caused by considering only a short time frame. We are considering longer or adaptable time sequence modeling to improve the encoding for gestures. In addition, in this work, we focused on maximizing flexibility in the representations the network learns to study patterns that emerge. Using pure convolution layers may retain more temporal constraints and future work could explore the differences in the representations learned by different network architectures.

While we expected adding more tasks would improve the classification results across all gestures, this is not necessarily the case. We observe from Table 3-4 that classification of Knot-Tying gestures improves when all the tasks were used in training, but classification of Needle-Passing gestures worsens. In the JIGSAWS dataset, each trial of a task is very consistent, with no variation in camera or phantom placement. The network may overfit on a particular setup and generalize badly when the camera and phantom placements change. More gestures are also present when multiple tasks are considered, leading to a harder classification task. More work is needed to determine what causes the drop in accuracy when using more tasks.

One limitation of our work is that, since the JIGSAWS dataset has a single camera perspective in all the videos, we do not account for different camera matrices. As the correspondence of the video is dependent on the camera view, there would need to be further consideration on how this pairing changes given a different camera matrix. The instrument pose should be reported with respect to the robot base so it should not be affected by different camera views. The network should learn a many-to-one mapping of the scene, where many different views of the scene could have the same kinematics. It would be an interesting experiment to see the resulting representation if we train it with different views of the same action. If the network could learn to generate similar representations of the same gestures across different views, the representation could be more robust to new tasks. The cost would be that the learning is less efficient and would require many samples to cover the range of potential camera poses.

Another way to address different camera views would be to use the known camera pose to warp the image to a base pose. This becomes difficult if there is a lot of camera motion and the overlap of the images at the two poses is small. This might benefit from combining with a soft-tissue model, such as described in the Chapter 4. Instead of directly using an image to kinematics mapping, we could use the image to update how the model is deforming and formulate a model-deformation to kinematics mapping.

Another limitation of this work is that our encoder network only considers the left camera feed rather than both streams of the stereo camera. Considering both could possibly lead to learning information on new tasks such as depth perception and also simultaneously improve accuracy on the currently existing tasks. Finally, while our work does classify surgeon skill, it does not address the specific differences in surgical behaviour (e. g., differences in kinematics) that could potentially address the gap in surgical skill and be used as milestones in surgeon training. Future work could also consider adding force estimation, such as through the method presented in Chapter 2, as input as applied force is correlated with skill.

This work has focused on demonstrating visually the separation in the representation learned through unsupervised learning. Its quantitative results lag behind the state-of-the-art supervised methods. While we focus on using the representation to do skill and gesture recognition as a proof of concept, there also are potentially more fitting applications that future work could explore, such as trajectory prediction and personalized feedback on the gesture level. Training these representations on a sufficiently large and diverse surgical dataset can potentially lead to a standardized architecture for parsing surgical activity that is highly transferable across datasets and tasks. One advantage of regressing a real-number representation of gestures over classification tasks is that we can measure distances between gestures. We can use it to perform fine-grained comparisons between surgeons at different skill levels performing different gestures. It is difficult to gather a sufficient dataset to train classifiers to distinguish each gesture at each skill level. Even with our current gesture classifier without skill considerations, we observe that some gestures have few samples and are therefore often mislabeled. The proposed representation does not learn from labels and thus does not suffer as much from the lack of samples in a particular class. It does show separation in skill however and could be used to evaluate how similarly one surgeon performs a gesture to other surgeons at various skill levels for fine-grained feedback of each surgeon's skill on each gesture. This allows us to evaluate not only whether a surgeon can perform a whole task well, but which gestures they may need to practice more.

Future work could explore including images as input. While we use optical flow since it contains the relevant information to map to the kinematics, a different learning task that makes use of images could retain more information about the scene. While the goal of the encoder-decoder structure is to extract information distinct from the medium of the input and output, we see that the input does affect the learned representation. Optical flow does well in representing large tissue motion such as the gesture "pushing needle through tissue". Its representations of tasks that have limited motion such as "orienting needle" are less distinct. As a result, the classifier correctly labeled the former much more often than the latter.

Another way to use the image could be to use it to classify pixels. For example, it can assign different labels to different parts of the tissue. While we can obtain an estimate of the PSM pixels by forward kinematics, we can also use computer vision to sharpen boundaries of the instrument given uncertainties in kinematics. Many actions in the JIGSAWS dataset are specific to the left hand, the right hand, or both. Knowing which hand caused the motion in the image would help distinguish between those actions. Future work could also consider more complex datasets where different tools are used. Extensions to this work could consider encoding both a label for which instrument or background each pixel belongs to, and the instrument type.

In addition to generalizing across different datasets, we envision that this self-supervised method can update its representations while any new user is performing tasks so that the classification results would improve based on real-time observations. This could in turn enable a universal technology that tracks surgical progress in real-time, giving feedback regarding possible mistakes, surgical scene depth, next gesture suggestion, etc. with a high accuracy. This generalized representation of surgical actions could turn the robot from an extension of the surgeon's arms to an intelligent assistant in the operating room.

Chapter 4

Interpreting scene dynamics

Robotic surgery has changed the way many surgeries are performed. Computer-assisted interventions are increasingly adopted for different procedures, yet clinical plans are developed on pre-operative images that are often hard to register with the surgical scene once tissue starts deforming. One approach is to use an intraoperative imaging modality, such as ultrasound, that can track the deformation in real-time and then be used to "warp" the preoperative image to match the deformed anatomy. Mohareri et al. [109] demonstrated this approach clinically using transrectal ultrasound to track prostate deformation during robot-assisted surgery, thereby enabling overlay of a preoperative magnetic resonance image (MRI).

Alternatively, navigation systems can make use of partial observations [110] or track deformations through biomechanical models [111] and then register to the surgical scene to provide intraoperative guidance. The current gold standard for accurate soft-tissue modeling is the finite-element method (FEM). Unfortunately, FEMs are generally too slow to be used intraoperatively. While there are approaches that speed up FEM using pre-processing [112], multi-threading [113], and GPU [114], these have so far only supported a limited set of geometric shapes and interactions. Alternative methods for soft-tissue simulations that can run in real-time, such as ChainMail [115], have not been widely adopted, potentially due to their lower accuracy if the mesh is not optimized. Brunet et al. identified a target of reducing errors to within

5 mm for providing useful clinical guidance with augmented reality and soft-tissue simulation [116].

Not only do robots allow physicians to perform more complex surgeries, they also open the possibility for machine learning algorithms to provide aid, such as automating surgical subtasks. Obtaining data to train these algorithms can be difficult, including, but not limited to, concerns regarding patient privacy. An alternate source of data may be the simulators doctors must practice on before operating on patients. Unfortunately, many simulators have rudimentary physics and cannot accurately model large deformations. Thus, these simulators test surgeons on simplified tasks to train agility rather than on a full surgery. While surgeons are adept at generalizing from these tasks to the clinic, algorithms are limited by the data they are provided. More accurate simulators that are capable of modeling complex geometries and interactions are necessary to create realistic tasks. These can both benefit the surgeon and provide high quality data to train the robot to intelligently aid physicians.

To extend simulations to provide intraoperative guidance, precise soft-tissue simulation is required. Accurate material parameters are integral to accurate FEM simulations, though boundary conditions and geometric model also play an important role [110]. These are often difficult to measure in a phantom, and impossible to obtain in a clinical setting. Inaccurate models in the FEM simulations result in inaccurate predictions of tissue behavior. Given these limitations inherent in any model, we propose a neural network that can provide corrections to an FEM model in the presence of large deformations and inaccurate material parameters. The network is self-supervised as it can be trained from data available in any robotic surgery: endoscope video and robot kinematics. Recent works have recovered the 3D surface during surgery from the endoscope video [117], which provides the ground truth for our network.

This chapter explores using machine learning in combination with models for fast soft-tissue simulation that account for real-time feedback [118] and can run in real time. This is a first step of a framework that could be capable of life-long learning and adapt models to patients throughout a surgery. This can provide better feedback for users and serve as a basis for generating datasets for data-driven methods. Current surgical subtask automation research on soft-tissue manipulation is limited to simple 2D phantoms [119] or path planning algorithms with no tissue interaction [120]. The proposed system was implemented on the first generation da Vinci Surgical System with the open source controllers of the da Vinci Research Kit (dVRK) [28].

Contributions

This chapter presents one of the first biomechanically accurate soft-tissue simulators that is fully integrated with the dVRK framework. The first part consists of a data-driven approach to improve FEM simulations of soft-tissue deformation based on robotics vision and kinematics data[118]. The second part uses a network to mimic FEM simulation results to speed up soft-tissue simulations [121]. Together, these parts present a real-time soft-tissue simulator that can use intraoperative measurements to improve accuracy. Adnan Munawar contributed the Blender plugin described in Section 4.4.1.5 to visualize the deformations predicted by the network. The code for the Blender plugin can be found at `https://github.com/adnanmunawar/blender_socket_comm`. The code for the network learning parts can be found at `https://github.com/JieYingWu/simulator_network`.

4.1 Previous work

4.1.1 Background

While FEM is widely adopted, mesh-less algorithms can be used for difficult-to-mesh cases [122]. Another alternative to FEM is to use neighborhood-aware cell neural networks modeled after the propagation of an electrical signal [123]. Since we focus on modeling one base phantom well, we assume that a mesh is available. This is generally the case in clinical scenarios, as there is a wealth of 3D meshes of organs which can be adapted to a patient-specific case based on a preoperative 3D image. On the other hand, mass-spring models [124] and ChainMail algorithms [115] can achieve real-time soft-tissue simulation; however, to obtain high enough accuracy for clinical use, they require optimizing the geometry of the mesh which limits their generalizability. We limit our review here to FEM and deep learning methods, and refer readers to [125] for a more general review of soft-tissue simulation methods. Li et al. use a particle filter with organ models to track tissue deformation [21]. Pfeiffer et al. use a U-Net architecture [126] to fill in the deformation of the entire organ given the deformation on a partial surface [110]. They show that their synthetically trained network can generalize to real cases, both in phantom and human CT data, with different geometry.

We first review the simulation platforms that exist for the dVRK. Second, we review the most relevant learning algorithms that have been proposed to improve and speed up simulations.

4.1.2 Simulators

Many existing works on simulators for the dVRK have focused on the robot kinematics. Gondokaryono et al. uses Gazebo to support environment objects like placing a camera but does not actually interact with any object [127]. This work was extended to include interactions by Munawar et al., which introduced a new simulation environment called the Asynchronous Multi-Body Framework (AMBF) [128]. The AMBF simulation environment uses a new robot description format and supports soft bodies. The aim here is rapidly prototyping complex environments for surgeon training rather than accurate physics. Fontanelli et al. uses V-REP to create a rigid body simulation of the da Vinci but does not support soft-tissue interaction due to V-REP's limited physics backend [129]. They discuss the desirability of a soft-tissue simulator and suggested using the SOFA framework as the physics simulator. The SOFA framework [130] focuses on simulation for medical purposes and has been validated on a variety of tasks from surgical training to guidance [131]. In this work, since we are interested in modeling the soft-tissue interaction, we chose SOFA as our physics simulator.

4.1.3 Learning from simulators

While FEMs are the standard today for accurate, deformable tissue simulation, other models have been proposed and the problem of parameter estimation is common. Bianchi et al. learned parameters of simulations by approximating soft tissue as spring-mass models [132]. They found that a system with homogeneous stiffness is insufficient to model the complexity of soft-tissue behavior.

More recently, researchers have begun training deep learning models to predict deformations from FEM models. Morooka et al. trains a network to predict the deformations of an FEM, where the input is a force vector and contact point, and the output is every point of the deformed mesh [133]. The size of the simulation object must be strictly reduced since their network is fully-connected and thus the network size scales poorly with input and output size. They use principal component analysis (PCA) to keep their network size tractable. Although their method could be trained end-to-end by using more recent architectures such as autoencoders, these are currently shown to provide limited advantage over PCA to model mesh deformations [134]. Haferssas et al. uses domain decomposition [135] to speed up computation.

Other works have used deep learning to predict deformation over time steps larger than for which the FEM is generally stable. Meister et al. presents an alternative approach to use a fully-connected network to predict the solution of the Total Lagrangian Explicit Dynamics needed for FEM for each vertex of the mesh. They show that this is stable for larger timesteps than those at which FEM is stable and could be used to speed up deformation [136]. Using convolutional layers allows networks to process larger meshes. Mendizabal et al. use a 3D U-Net architecture to learn the deformation of a mesh given forces represented by a 3D grid [137]. They apply simulated force to a known mesh and generate the desired deformation using FEM. Their network aims to predict the FEM deformation.

While these works have generally used voxel-based convolutions, graph-based neural networks [138] could be more topography-aware than voxel-based ones. Since meshes are modeled as tetra- and hexahedrals, they naturally have a graph structure. Graph-based neural networks have been used to learn propagation in particle-based simulators [139, 140]. Sanchez-Gonzalez et al. encode particles as graphs and use message passing to learn the acceleration of each simulation step [140]. While these show the potential of graph networks to simulate deformations, they consider plastic deformations only. Elastic materials such as those considered in this work present different challenges. On the one hand, the forces acting on a node not only depend on external factors like gravity and on the node's collision with its neighbors, but also on each node's initial position. On the other hand, the edges are predefined and do not generally change. This avoids the need for dynamic graph building as described by Li et al. [139].

4.2 Data collection

4.2.1 Physical setup

Fig. 4-1 shows the phantom setup. The phantom was created by pouring liquid plastic into a mold with some amount of hardener and softeners from M-F Manufacturing (Fort Worth, TX). After the phantom has set, we cut out a rectangular block. After trimming, the phantom's dimensions are $68.7 \times 35.8 \times 39.3\,\text{mm}$ and it weighs $104.01\,\text{g}$. We do not measure the stiffness of the phantom but perform a parameter search for its material parameters.

The robot instrument is positioned to the top right and the depth camera is on the left, mounted above the workspace. We move the robot instrument to interact with the phantom while capturing its Cartesian positions through ROS [78]. We move a PSM directly, without teleoperation, to interact with the phantom while recording its kinematics and measuring the deformation of the phantom using an Intel Realsense SR300 camera (Intel, Santa Clara, CA). Intel reports the accuracy of the depth measurement to be 2 mm. Interactions consist of probes to the top and sides of the phantom. Since the plastic is translucent after setting, we coat the phantom in spray paint and potato starch so that it can be measured by the depth camera. Each frame of the depth camera is read out as a point cloud. We use the Point Cloud Library (PCL) [141] to remove points from the instrument and the table, as well as outliers so that the only points that remain are the ones from the phantom. The scene is primarily composed of three objects: the table, the phantom, and the robot instrument. We cluster the points by a manually identified distance threshold. This was sufficient to separate the points into three groups, with each group belonging to one of the objects in the scene. As the phantom is the second largest object in the scene, our segmentation algorithm selects the second largest group of points as belonging to the phantom and removes all other points. Then we subsample the points to about 45k points, or about 16.5 points per mm^2.

We capture 12 sequences of interactions between the instrument and the phantom. Robot position is captured at 1 kHz while the depth image is captured at 30 Hz. We subsample the robot positions to match that of the depth images in time. This results in around 14 minutes, or 25k frames, of video. Of those interactions, the first one is 2 minutes and the rest are around 1 minute. The 2 minutes, 3794 frames, clip is used as the validation set. The next two one-minute segments, 4417 frames, form the test set. Each sequence contains several distinct interactions with the phantom. The tool is lifted to not touch the phantom between interactions.

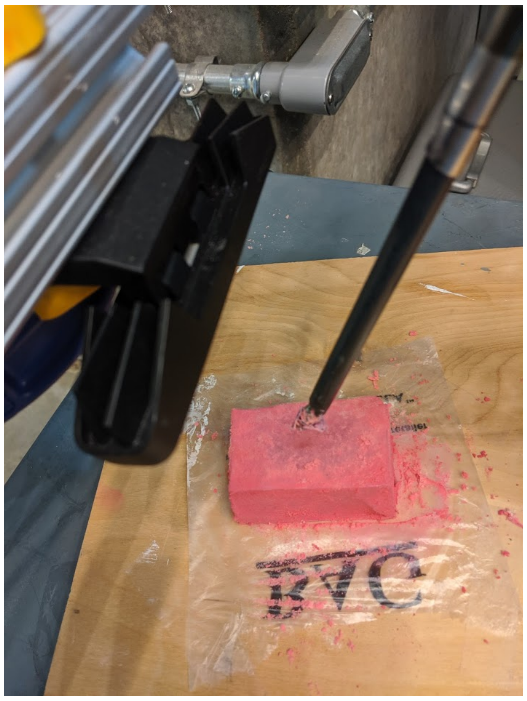


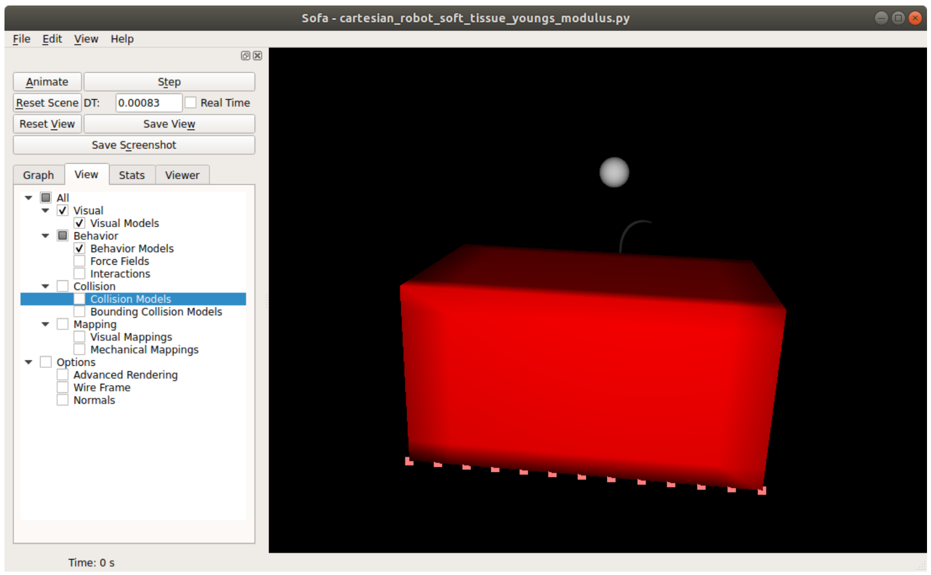
Figure 4-1. Deformable phantom setup. The RGBD camera is mounted above the workspace on the left. The phantom is coated with spray paint and potato starch to improve its visibility to the depth camera.

4.2.2 Simulation scene

We create the surface mesh of the phantom in Solidworks (Dassault Systèmes, Vélizy-Villacoublay, France) and fill it with Gmsh [142] to create a solid, tetrahedral mesh. SOFA provides different templates for modeling objects; here we focus on two of them. With the 'Rigid3d' template, objects have 7 parameters (3 translational and 4 for rotation quaternion) while with the 'Vec3d' template, each vertex of the object's mesh can be individually set. We use Vec3d for the phantom and Rigid3d for the instrument.

While we implemented the robot in SOFA, computing the constrained movements was too computationally expensive and not necessary for learning the mesh deformation. Instead, to simplify computation, we only simulate the end-effector pose and approximate it as a sphere with a 5 mm radius. The sphere is moved based on the Cartesian position of the end-effector at each step. Additionally, to avoid simulating computationally expensive friction interaction with the table, we suspend the phantom by fixing the bottom vertices in space rather than placing it on a plane. The fixed vertices are marked by pink cube overlays in Fig. 4-2. Gravity is currently not simulated. We use a parameter search to find the best material parameters for the phantom. The results are shown in section 4.2.4.

We calibrate the real robot to the simulator reference frame. We capture a calibration sequence where we move the robot to touch each of the four corners of the phantom and extract the corner location from the robot and then perform a rigid registration to the corresponding points of the simulated phantom. The simulated robot replays the calibrated actions and the simulation saves the positions of the vertices of the phantom mesh at each timestep that has a corresponding point cloud label. At each timestep, the robot end-effector position is updated based on the captured kinematics from the interactions of the physical robot. To ensure the FEM simulations run stably, the simulation timestep is much smaller than the rate at which the robot kinematics is captured. We linearly interpolate the robot end-effector position for simulation timesteps where we do not have a corresponding label.


Sofa - cartesian_robot_soft_tissue_youngs_modulus.py

DT: 0.00083

Time: 0 s

Figure 4-2. Simulated phantom. The gray sphere is the end-effector. The red block is the deformable phantom, with the pink cubes showing where the phantom mesh is fixed.

The model in SOFA is represented as a solid, tetrahedral mesh of size $13 \times 5{\times}5$. Even with the small mesh, it takes on the order of hours to simulate a minute of interactions. This may be improved by multi-threading but as that does not improve throughput over running multiple instances, it was not implemented for this work. A finer surface mesh is attached to that for smoother visualization. The da Vinci instrument end-effector is approximated as a sphere with comparatively large mass of 1000 g. We set the minimum contact distance to be 0.5 mm and use Euler implicit solver with Rayleigh stiffness and mass both set to 0.1.

4.2.3 Calibration between physical and simulation scenes

To register from the camera to the simulation, we extract the point cloud of the phantom without any robot interaction. We find the transformation between the phantom and that point cloud using iterative closest point (ICP) [143] as implemented in PCL. Since the top of the block is flat and featureless, we manually initialize the registration so that the algorithm registers to the correct pose. To calibrate between the robot coordinate frame and the simulated scene, we manipulate the robot so that its end-effector successively touches each of the phantom's corners. We can then calculate the rigid transformation between the two scenes by the four corresponding points.

4.2.4 Material parameter search

The main parameters of soft tissue can be considered to be its Young's Modulus and Poisson's Ratio. Uncertainty in the parameters come from the unknown mixture of softeners and hardeners during construction of the phantom as well as its age. To limit the search space and since the two parameters are interdependent, we focus our parameter search on the Young's Modulus. Values for the Poisson's Ratio are more consistent in literature to be in the 0.4-0.5 range, whereas estimates of the Young's

Modulus vary anywhere in the range of 1e-1 to 1e4 kPa depending on the specific construction of the phantom. We set the Poisson's ratio based on previous work [144].

To perform a grid search for the optimal Young's Modulus, we choose one interaction sample from the training set, about 1 min in duration, and simulate it with different Young's Modulus. Since FEMs are computationally expensive to run, we separate the test in two stages. First, we sample the Young's Modulus at every factor of 10 from 1e1 to 1e6 kPa to explore a broad range of values. We write out the simulated phantom positions, super-sample the mesh by a factor of 3, and compare the vertex positions to the measured point cloud. The average distance from a point on the depth camera point cloud to the simulated mesh are reported in Tab. 4-1.

Table 4-1. A one-minute interaction with the phantom is simulated with each of the Young's Modulus (kPa) and the average distance in mm from measured point cloud to the mesh vertices is reported.

Young's Modulus	10	10^2	10^3	10^4	10^5	10^6
Error (mm^2)8.4981	6.0760	5.5417	5.2632	5.1847	9.3356	

Next, we pick the optimal range and explore that more finely, in this case around 1e3 to 1e5 kPa. We test the interval between 1e3 to 1e4 at every 2.5e3, and the interval between 1e4 and 1e5 at every 2.5e4 kPa. Results are shown in Tab. 4-2.

Table 4-2. A one-minute interaction with the phantom is simulated with each of the Young's Modulus (kPa) and the average distance in mm from measured point cloud to the mesh vertices is reported. The simulation with Young's Modulus of 5e4 kPa did not converge.

Young's Modulus	2.5×10^3	5×10^3	7.5×10^3	2.5×10^4	5×10^4	7.5×10^4
Error (mm^2)	5.2724	4.9041	5.5025	5.2202	N/A	5.4565

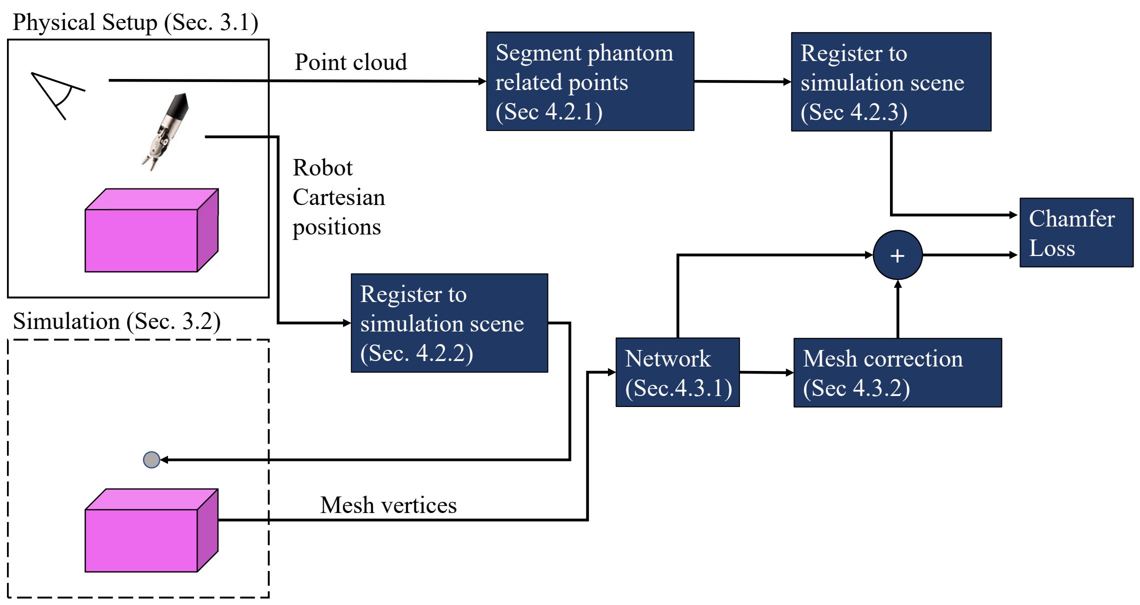
We observe that Young's Modulus optimization is not convex, potentially due to measurement error since point clouds are generally noisy, and that a small change in parameters can lead to a result that did not converge. This reinforces the need to find a way to learn the optimal parameters rather than search exhaustively. 5e3 kPa was selected to be the optimal Young's Modulus. We simulate all the interactions using 5e3 kPa to generate the input to our network.

4.3 Method for correcting soft-tissue simulations

Our goal is to learn correction factors for an FEM simulation by comparing the simulation results to a physical setup. To that end, there are three main components to our setup. First, we create a physical phantom and use the da Vinci patient-side manipulator (PSM) to palpate the phantom and capture the RGBD video and robot kinematics. Second, we create the same scene in simulation and replay the interaction to simulate the shape of the phantom as it deforms from the robot interactions. Lastly, we compare the simulation mesh vertex positions and the measured position of the physical phantom and train a network to correct the simulated positions. Fig. 4-3 shows an overview of the setup.

4.3.1 Network architecture

We train the network using the pre-calculated FEM-simulated phantom mesh vertex positions as input. The output is a correction factor for the vertex positions to match the real, measured positions. The vertices of the phantom mesh are represented by a 3D matrix, where each element has three values for the 3D position of that vertex. We follow the architecture proposed by previous work and use a 3D-UNet [126] to process the mesh. The network contains 3 blocks, each with 2 convolutional layers with kernel size of 3 and feature size of 64, 128, and 256 respectively. There is a Max-pooling layer between each block with kernel size of 2 to downsample the mesh. We keep the default activation function, ReLU. The kinematics information is inserted at the bottom of the network, where the spatial information is the most condensed. The robot pose is concatenated as additional features to each of the voxels. This is inspired by the work of Finn and Levine [145] in robot-motion planning where the action is inserted



Physical Setup (Sec. 3.1)
Point cloud
Segment phantom related points (Sec 4.2.1)
Register to simulation scene (Sec 4.2.3)
Robot Cartesian positions
Chamfer Loss
+
Simulation (Sec. 3.2)
Register to simulation scene (Sec. 4.2.2)
Network (Sec.4.3.1)
Mesh correction (Sec 4.3.2)
Mesh vertices

Figure 4-3. Overview of the proposed system. The left blocks show the physical setup and its corresponding simulation scene. The system captures the robot Cartesian positions as it is moved and registers them to the simulation scene, where it is replayed. The mesh vertices from the simulation are read out and fed into a network which predicts a correction factor. It is trained by comparing the simulated mesh plus the correction factor with the point cloud captured from the physical setup.

in the middle of the video processing pipeline so the first layers focus solely on images. We add 4 convolutional layers, also with kernel size 3, to incorporate the kinematics information. Since the kinematic information is added as features, we also use them to reduce the feature size back to 256 so that the feature sizes match when concatenated with previous layers for the decoder.

Here, we want the network to extract global mesh information before changing the portion of the mesh with which the robot is interacting. The end-effector position is concatenated to each of the voxels before the decoding layers. The network learns which part of the mesh the end-effector affects and produces a correction step for the displacement of each vertex so the output is the same size as the input. The correction is added to the mesh and we compare this to the point cloud captured by a depth camera for ground truth displacements.

The depth camera only provides labels for the top layer of the phantom. Therefore, we only allow the correction to be applied to the top layer of the mesh. We also test a 2D version of UNet where only the top layer of the mesh is passed to the network, shown in Fig. 4-4.

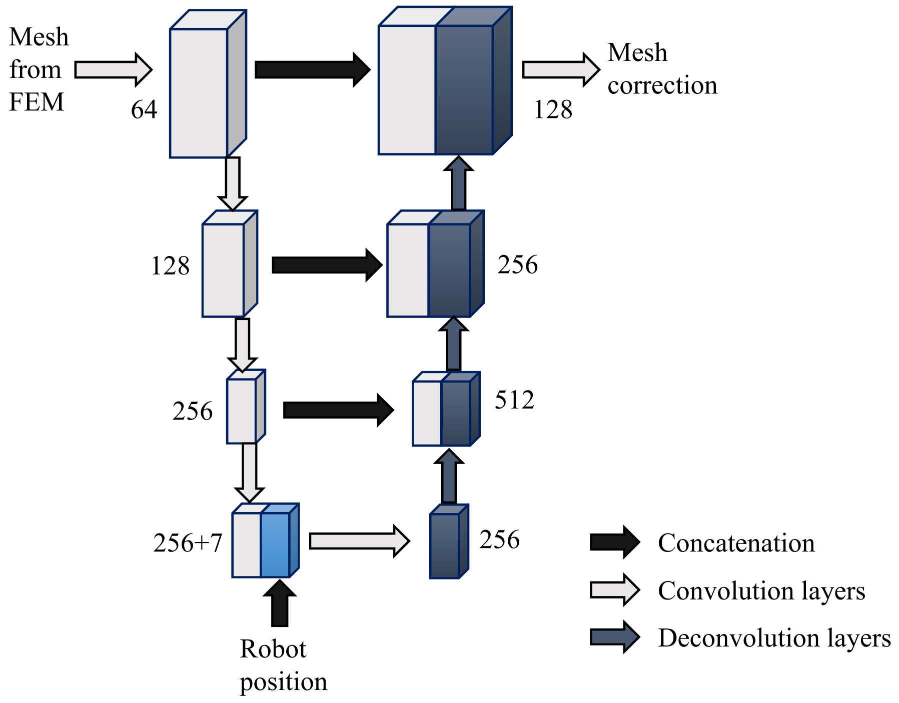


Figure 4-4. The network architecture for calculating the mesh correction. We modify the 3D-UNet [126] to insert the kinematics at the bottom, as shown by the light blue block. The robot position is concatenated to every voxel of the mesh as a feature. Each block features two convolutional layers and one Max-pooling. The numbers by each block indicate the number of features. All convolutions are with kernel size 3 and padding 1.

4.3.2 Training from observing interactions

At each simulation step, we pass the position of the phantom and the robot position as input to a neural network. The network predicts a correction factor and updates its weight by comparing the corrected mesh and actual phantom position. To prevent vertices from crossing, the maximum correction in each dimension is scaled to be half of the voxel length in each direction. The training loss is the Chamfer distance, or the mean shortest distance between one point on a point cloud to another point cloud. In this case, we represent our mesh as a point cloud of its vertices. We initially trained with Hausdorff distance but since it is sensitive to outliers, the training did not converge well. Using the network to calculate a displacement map rather than the positions directly avoids the need for formulating a regularization term in the loss function to prevent the vertex positions from crossing over each other. This also gives us an additional factor to control how much we want to rely on the simulation. If we trust our FEM more, we can set the maximal displacement to be lower, or higher if we are more uncertain about the FEM parameters.

During test time, at the end of each timestep, the simulator passes the position of the phantom mesh to the trained network to get the correction. In the 2D network case, we only pass in the top layer of the mesh to the network. In our training, we noticed that since our ground truth can only constrain the top layer of the mesh, only those values are valid and are used to update the simulation. The other nodes stay as the simulator calculates them.

To compare the mesh to the point cloud, we super-sample the mesh by a factor of three in all directions using linear interpolation. The mesh must be fairly sparse due to computation power constraints for the FEM simulation but this leads to inefficient learning for the network as the loss is not smooth. This does not change the form of the mesh but provides more points for the point cloud to point cloud matching. Using this point cloud loss avoids the more computationally expensive point-to-surface matching. We only calculate loss from the direction of the depth camera output to the mesh since there may be no correspondences in the other direction due to occlusion from the surgical instrument.

4.3.3 Results for network correction

We train the network on 9 min of the captured video samples, or 15 872 samples, until convergence. Convergence is measured by loss on the 2 min of validation data flattening or increasing. We test on the remaining two sequences of about 2 min. The mesh is refined before calculating the Chamfer loss to provide a smoother loss for the network to train on. We compare the performance using 2D and 3D networks on two sequences as shown in Table 4-3. One interesting use case for the network is when the FEM parameters are sub-optimal. Since the parameter search is time-consuming, we test if the network can correct for non-optimal parameters. We run the simulations with the same setup, but set the Young's Modulus to 1e4 kPa and 1e1 kPa. Then we train and test the network using the same data split as before.

Table 4-3. The loss from FEM directly and after correction using 2D and 3D networks. The loss is the average squared distance between each point on the point cloud to the closest super-sampled mesh vertex in mm. The simulation was run with different Young's Modulus (kPa).

		No network	2D network	Improvement	3D network	Improvement
5e3	Seq. 1	5.4627	4.3025	21.2%	4.3234	20.9%
	Seq. 2	5.1170	4.3450	15.1%	4.6269	9.6%
1e4	Seq. 1	5.2333	4.2323	19.1 %	4.6446	11.2%
	Seq. 2	5.2734	4.3746	18.4%	4.9828	5.5%
1e1	Seq. 1	8.2322	5.6235	31.7%	5.9614	27.6%
	Seq. 2	9.2899	6.4518	30.6%	6.7030	27.8%

4.3.4 Discussion

We show a first step towards a framework to train networks to learn the behavior of soft-tissue directly from robotic surgery data. This approach combines model-based simulations with real-time observations to correct for inaccurate simulation parameters. Using the surface deformation and a robot position, we trained the network to improve FEM results by 15-30% over a range of simulation parameters. This network can be used to improve FEM results where we do not have reliable material parameters and implicitly adjust for boundary conditions and other factors that are often intractable to model. We show that using the proposed correction factor, even starting from a

Young's Modulus 5×10^2 kPa away from the optimum that we found, the network reduces the error to within 1.5 mm of the most accurate uncorrected simulation.

From our parameter search, we see that the best Young's Modulus found on one video sequence is not necessarily the best for other sequences. There are unmodeled forces that affect deformation. Our proposed data-driven method avoids the need to search the high-dimensional space for all these factors. This observation reinforces the advantage of an observation-based correction. Additionally, heterogeneous tissue is generally harder to model than homogeneous tissue and represents further opportunities for data-driven corrections. Since the learning could be done online, this network can be updated during a procedure if the models do not match real observations. The patient's soft-tissue characteristics may change during surgery and our method could be adapted to do life-long learning, correcting for unmodeled changes.

We expected the 3D-UNet to provide better correction than the 2D, since the network may take into account the position of the next-to-top vertices; however, the 2D network performed better. This may be due to not using enough training data as the 3D network has more parameters to train and not enough constraints for the non-top layers of the mesh. The 3D network may benefit from integrating synthetic data and using the FEM to provide constraints for the unobservable vertices. This would provide ground truth for the entire mesh rather than only the surface.

We currently do not include gravity, boundary conditions, or the velocity of the end-effector, which may improve baseline simulation and the learned correction factors. Future work could incorporate this information and even add additional sources of information that we currently cannot measure. One example is the applied force, which we cannot measure but we can obtain through the force estimation algorithm proposed in Chapter 2. We use default hyperparameters in our network, and a careful search may improve results. While the network currently uses the FEM output, in the next section, we consider learning the simulation step with a network to achieve real-time deformations. Training on FEM results inherently limits their accuracies to be that of the FEM and it would be interesting to incorporate data from physical interactions.

4.4 Network-based soft-tissue simulation

The goal of this section is to predict successive update steps in the manner of a traditional simulator, such as an FEM. We start with the 3D mesh of the object. We construct a network that takes the mesh and the robot's kinematics to estimate an update step [121]. The update step is applied to the mesh and the updated mesh is passed with the next set of robot kinematics to the network for it to predict the next update step, as shown in Fig. 4-5. The output is the update step for each node, which is added to the current mesh vertex positions. The sum should match the next step of the FEM. For stability, the update in each step is limited to be at most one voxel length, based on the mesh element's size, in each direction.

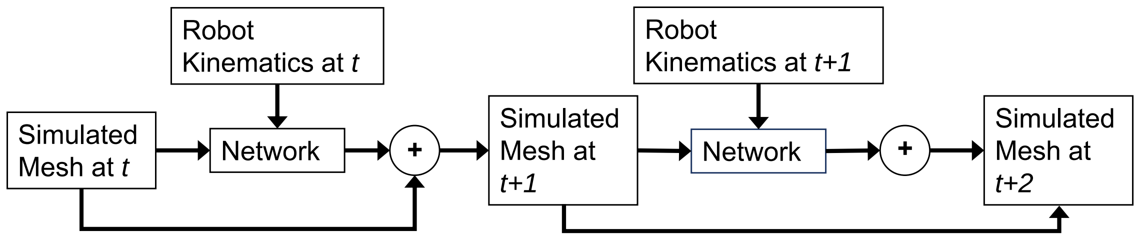


Figure 4-5. Overview of the proposed system. The network takes the current mesh and robot kinematics and predicts an update step. The update step is added to the current mesh and the sum is passed to the network again, with the next robot kinematics, to do step-wise simulation.

4.4.1 Network input

The input to the network is a graph where each node of the graph corresponds to a vertex on the mesh. The features of the node consist of the vertex's relative position concatenated with information on the robot kinematics.

4.4.1.1 Mesh

The graph representation of the mesh consists of the relative node positions and edges. The edges are defined by the mesh. The relative node positions are represented by the differences between each node's current position and its position in the base mesh. Intuitively, this encourages the network to learn the physical property of elasticity without needing to implicitly learn where each node's resting position should be. Without an external force, the mesh should return to its original shape. Using the offset rather than absolute position of mesh nodes may also help with learning a generalizable convolutional kernel.

4.4.1.2 Robot kinematics

For each node in the network, we extend its feature representation with robot kinematics features. These kinematics features are initialized as zero for all points of the mesh. Then, if the robot is within a threshold of the mesh, we set the kinematics features of the closest node to the position and velocity of the end-effector. For our setup, we empirically determined the threshold size to be 5.5 mm based on the instrument tip. The velocity of the end-effector is computed independently of the position based on the method proposed in [77]. How much the mesh changes is mainly dependent on how the robot is pushing against it. The more the robot moves, the more the mesh should deform. Since we limit our consideration to soft-tissue deformations, the deformation is largest where the robot touches the mesh and radiates outwards from there.

4.4.1.3 Data augmentation

We use a data augmentation strategy that aims to capture the network error. Once the errors of the network converge in predicting the next FEM update, we introduce the following augmentation to the input. For 50% of the input, rather than loading the mesh at time t, the network instead loads a simulated mesh. The simulated mesh is constructed by applying the kinematics and FEM mesh from $t-1$ as inputs to the network from the last epoch. This is then used as the input to the current network to predict the next update step. Fig. 4-6 shows how the simulated mesh is generated and used as input for training the network.

The network predicts the next simulation update and its output is added to the simulated mesh. The sum is compared to the FEM output at the $t+1$ step. This step is important to prevent simulation errors from accumulating over time steps. Since the FEM should simulate step-wise deformation of the tissue, its inputs at test time would diverge from the training samples if its training data is drawn only from the FEM. By incorporating the network simulation in data augmentation, the network learns to correct for errors it introduces.

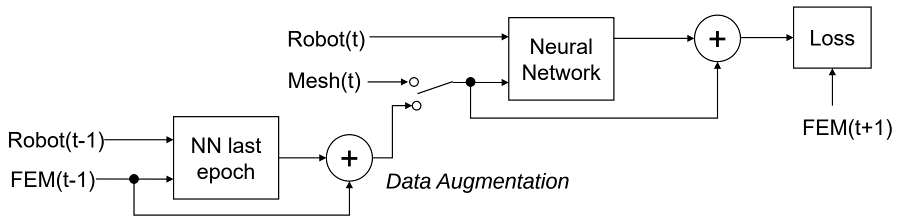


Figure 4-6. Data augmentation strategy; once network is sufficiently trained, 50% of input uses simulated mesh predicted from previous FEM mesh using network from last epoch.

4.4.1.4 Network architecture

We use a U-Net architecture as it has been shown in the literature [110, 137] to work well in estimating mesh deformations. Unlike previous works, we use graph-based convolutions as we observed that the voxel-based network accumulated errors at the edges of the mesh due to padding. The network architecture, shown in Fig. 4-7, is based on the method described in [138]. We use ReLU for internal activation functions and sigmoid for last layer activation. The output should have the same number of

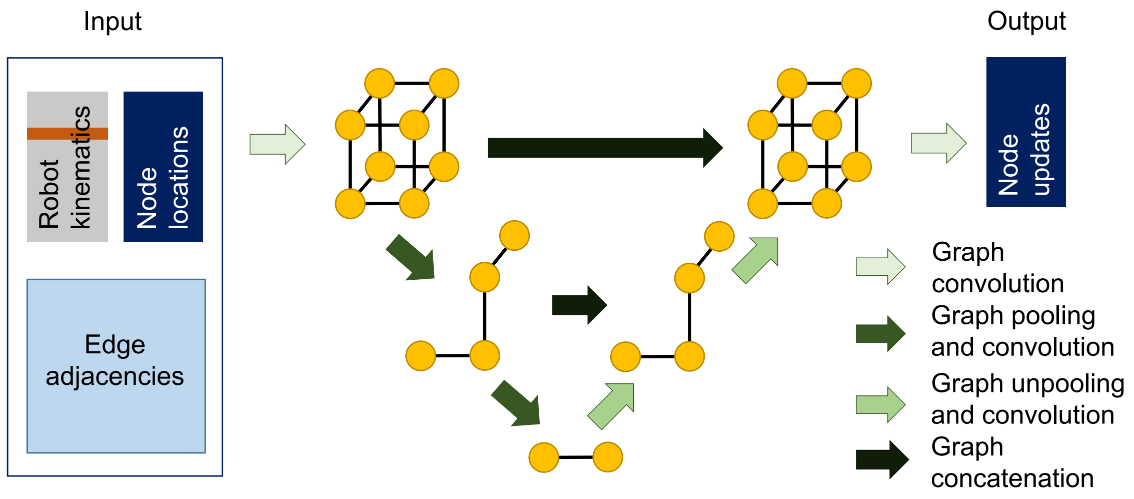


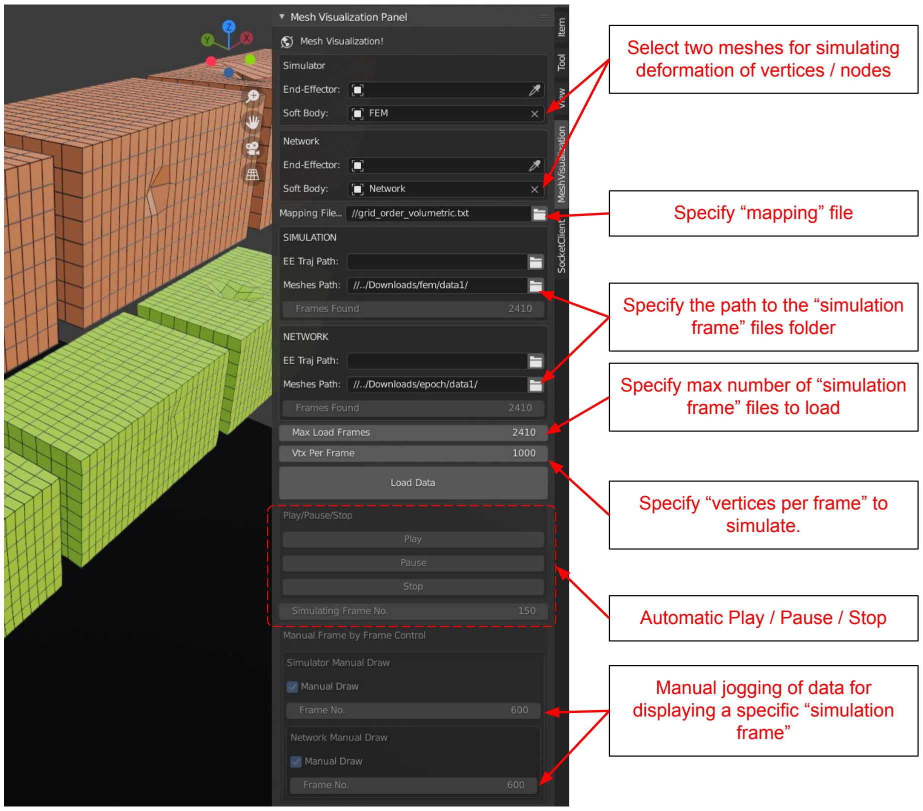
Figure 4-7. Simulator network architecture. The input consists of three parts. The node locations and edge adjacency are given by the mesh structure. While the node location is updated at each step, the edge adjacencies stay constant. The robot kinematics features are zero everywhere except where the robot end-effector is closest to the mesh, where it is the end-effector's position and velocity (indicated by the orange stripe). The network involves convolutions over the graph structure and pooling and unpooling between layers. Each of the network's hidden layers has 256 features. The output consists of three features for each mesh node to update its position for the simulation step.

nodes as the input and have feature size three representing the update step. The output of the sigmoid is shifted to be zero-centered and scaled based on the voxel size.

4.4.1.5 Visualization

We implemented a visualization client in Blender [146] to show the similarities and differences between the simulations. Blender is a free, open-source and versatile tool that can be extended via its plugin-based API for different applications. We used this API to develop a visualizer (*blender visualizer*) for our *simulation frame* files. The graphical interface of the *blender visualizer* is shown in Fig. 4-8.

Our custom visualizer was motivated by the fact that using existing simulators resulted in conflicts between the network updates and the simulator clock. We realized the importance of creating a portable package that could be used across different research projects. For this purpose, we first defined a simple data format for outputting


Item
Tool
View
MeshVisualization
SocketClient
Load Data
Play
Pause
Stop
Select two meshes for simulating deformation of vertices / nodes
Specify "mapping" file
Specify the path to the "simulation frame" files folder
Specify max number of "simulation frame" files to load
Specify "vertices per frame" to simulate.
Automatic Play / Pause / Stop
Manual jogging of data for displaying a specific "simulation frame"

Mesh Visualization Panel
Mesh Visualization!
Simulator
End-Effector:
Soft Body: FEM
Network
End-Effector:
Soft Body: Network
Mapping File... //grid_order_volumetric.txt
SIMULATION
EE Traj Path:
Meshes Path: //../Downloads/fem/data1/
Frames Found 2410
NETWORK
EE Traj Path:
Meshes Path: //../Downloads/epoch/data1/
Frames Found 2410
Max Load Frames 2410
Vtx Per Frame 1000
Play/Pause/Stop
Simulating Frame No. 150
Manual Frame by Frame Control
Simulator Manual Draw
✓ Manual Draw
Frame No. 600
Network Manual Draw
✓ Manual Draw
Frame No. 600

Figure 4-8. The *blender visualizer* plugin. Users can pick two meshes to simulate at once. The plugin supports both automatic playing of data as well as manual jogging by providing the specific *frame number* to simulate.

the soft-tissue deformation data. This data format requires a predefined volumetric mesh, which could be created from pre-operative scans. Then for each time step, the data format contains a sequential array of vertex / node positions at each simulation time-frame. Thus, for example, a simulation episode consisting of 30 seconds with 10 frames per second (FPS) would result in 300 text files, where each file is called a *simulation frame* file.

To map the indexes of the vertices / nodes in the *simulation frame* files and the mesh represented in Blender, we created another text file, called the *mapping* file. The format in this text file is a two column array, where the first column is the index of the vertex / node in the *simulation frame* files and the second column is the corresponding vertex index of the Blender mesh. Fig. 4-9 shows an example of how the mapping file is used to match the vertices from the network output to the Blender scene.

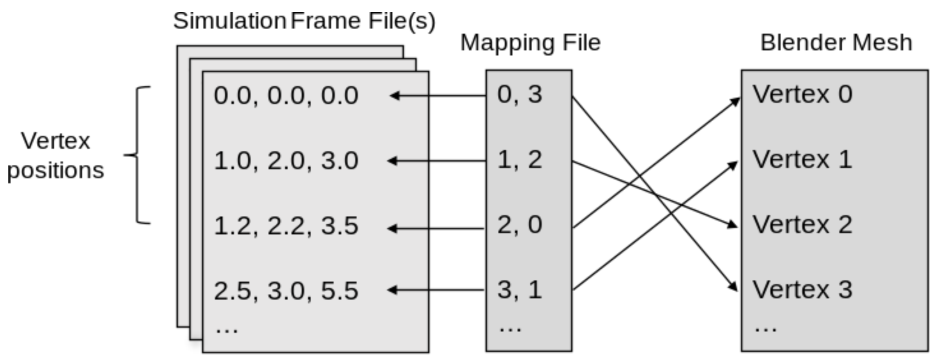
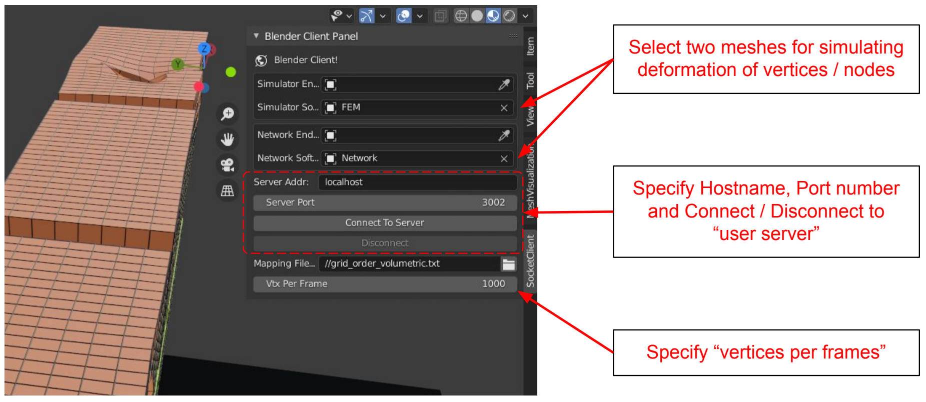


Figure 4-9. Example of using the mapping file to match vertices from network to Blender visualization.

To generate these mappings, we created another Blender plugin (*blender client*) that provides an easy to use socket interface for querying vertex positions and indexes of the Blender mesh. We complemented this with a Python package (*user server*) that abstracts the socket interface and provides the user with an API for querying or commanding vertices of the Blender mesh. The (*user server*) can process multiple vertices / nodes at the same time. Fig. 4-8 and Fig. 4-10 show the graphical interface of the *blender visualizer* and *blender client*, respectively. Fig. 4-11 shows the API exposed by the *user server*.


Blender Client Panel
Blender Client!
Item
Tool
View
MeshVisualization
SocketClient
Connect To Server
Disconnect
Select two meshes for simulating deformation of vertices / nodes
Specify Hostname, Port number and Connect / Disconnect to “user server”
Specify “vertices per frames”

Simulator En...
Simulator So... FEM
Network End...
Network Soft... Network
Server Addr: localhost
Server Port 3002
Mapping File... //grid_order_volumetric.txt
Vtx Per Frame 1000

Figure 4-10. The *blender client* plugin.

The *blender visualizer*, *user server* and *blender client* together allow for the visualization of any soft-body deformation data once it is ported to the *simulation frame* format. A useful feature of these packages is the support for real-time visualization of data. Blender's plugin API executes the callbacks in sequence with the graphics loop and as a result, any form of heavy processing within the callback method slows down Blender. The querying and updating of multiple vertices at once and processing the socket interface are a few such examples. To circumvent these issues, we implemented a combination of maintaining a shared data queue and populating it using separate threads. The plugin's callback method then reads a maximum number of vertex commands, specified by a user defined *vertex per frames* parameter, from the shared data queue. This allows for the real-time visualization of soft-tissue deformation. The source code of all these packages is provided [1].

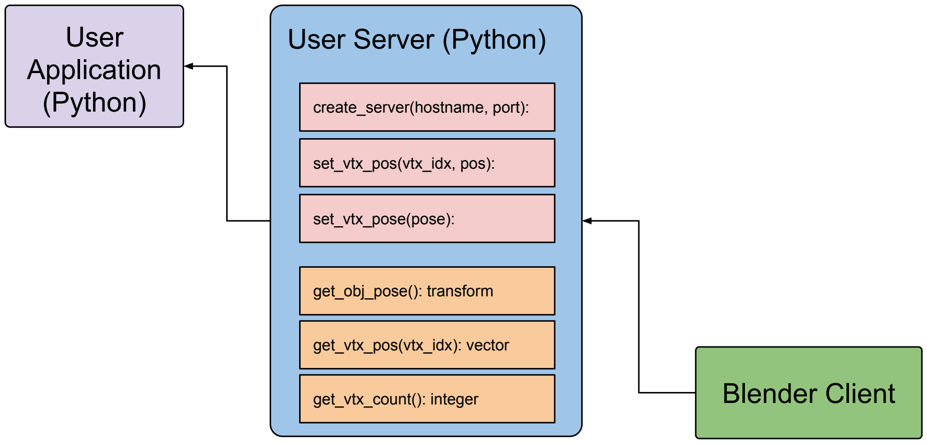

User Application (Python)
User Server (Python)
create_server(hostname, port):
set_vtx_pos(vtx_idx, pos):
set_vtx_pose(pose):
get_obj_pose(): transform
get_vtx_pos(vtx_idx): vector
get_vtx_count(): integer
Blender Client

Figure 4-11. The API of the *user server*.

This communication protocol allows us to use the neural network output to replace the physics backend and visualize estimated deformations in real time.

4.4.1.6 Implementation

This work was implemented on a workstation with a Xeon Processor E5-1630 v4 CPU (Intel, Santa Clara CA, USA) and a Titan V GPU (Nvidia, Santa Clara CA, USA). We implemented the network in Pytorch [85]. Since neural networks have been shown by previous work [136] to be stable over larger simulation steps compared with FEMs, we

[1]https://github.com/adnanmunawar/blender_socket_comm

sampled the output of the FEM to match the camera's frequency of 30 Hz to create the training data for the simulator network. We used L2 loss between the network output and the FEM ground truth. We chose stochastic gradient descent as the optimizer, setting momentum to 0.9 and learning rate to 10^{-4}, with decay on plateau scheduler. While the final network can update node positions by one voxel-step in each direction, to speed up convergence, we initially trained with each update step set to be at most half a voxel-step in each direction. Once that has converged, we increase the step size to one full voxel.

From the gel phantom dataset, we exclude the two sequences from the training set that were captured at a different frequency than the other sequences. While a coarser mesh of $13 \times 5 \times 5$ was determined to be sufficient in our previous work, we increased the size of the mesh to $25 \times 9 \times 9$ here as it resulted in the network learning more stable simulations. We empirically set the FEM simulation time step to 0.1 ms as a balance between computation time and accuracy of the FEM.

4.4.2 Results

We test the simulation network on the held out sequence, which is 80 s long and consists of several interactions with the phantom both on top and on the side. Fig. 4-12 shows simulation results from both the FEM and the network for samples from the test sequence.

Qualitatively, the network errors come from underestimating the deformations in some cases. Interactions to the side of the mesh may also introduce errors. We hypothesize that this is because interactions to different sides of the mesh have different effects and are thus harder to learn.

Although the entire sequence is performed as one simulation, we split the frames into 10 s segments for error calculations. This captures the range of errors resulting from different amounts of interaction with the phantom. In particular, the beginning

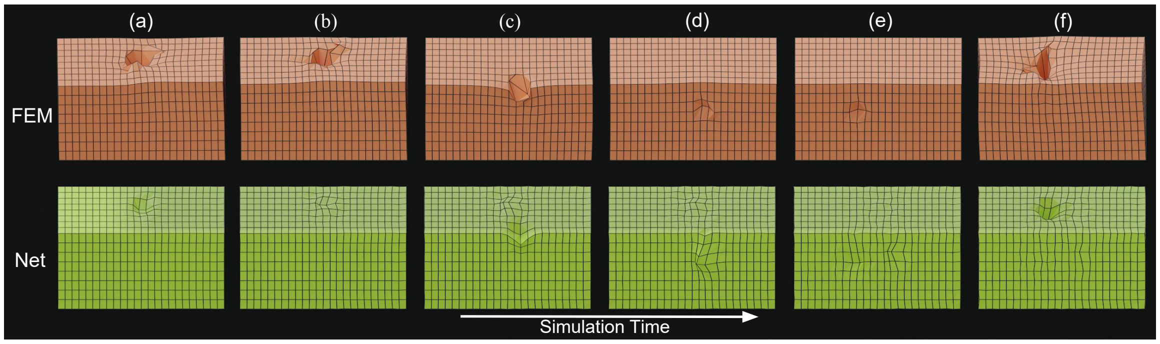


Figure 4-12. Sample frames of the simulation sequence over 80 s and multiple interactions with the phantom. The network generally matches the FEM predictions. Comparing frames (a) and (b), we see that the network initially detected the interaction but returned to the undeformed mesh too quickly. In frame (d), we see that there is an error in the direction of the deformation that is propagated through the remaining frames as the network is slow to recover from unexpected errors.

and end sequences may have fewer interactions. This results in 8 sequences. As Table 4-4 shows, the error between the FEM and the network prediction for each sequence, as measured by Euclidean distance, is small. It is much smaller than the distance from the point cloud to either the FEM or network simulation, which we measured as the average distance from each point in the point cloud to its closest neighbor on the mesh. This suggests that the error between the network prediction and the FEM is within the range of error from modeling using FEM. The advantage of the network simulation is that the total runtime to generate sequences with the network is 47 s. In comparison, the FEM took just under 15 h to run. The runtime is large for the mesh size as we use collision models between the mesh and the instrument, rather than directly applying forces on the mesh. Using a method for force estimation at the instrument tip, such as proposed in [32], and applying the force directly to the mesh could reduce the simulation time. A larger step size could also be used to reduce run time, but at the cost of larger error when compared to the real observations.

Additionally, we evaluate how incorporating real-time observations to correct the meshes would affect each case. We implement the 2D U-Net version of the correction network described in Section 4.3 and compare the error between the point cloud and

Table 4-4. Average (standard deviation) error in mm for each sequence between the FEM, the network simulation meshes, and the observed point cloud (PC). Error between FEM and network prediction meshes is measured as the Euclidean distance between each vertex. Point cloud to mesh error is measured as average distance of every point on the point cloud to the closest point on the mesh.

Seq.	FEM to Net	PC to FEM	PC to Net
1	0.51 (0.38)	4.56 (2.61)	4.48 (2.55)
2	0.60 (0.58)	4.15 (1.52)	4.00 (1.42)
3	0.54 (0.28)	5.85 (0.69)	5.63 (0.67)
4	0.65 (0.37)	5.31 (1.47)	5.05 (1.53)
5	1.04 (1.24)	3.48 (1.40)	3.19 (1.43)
6	0.81 (0.83)	3.97 (1.61)	3.88 (1.67)
7	0.86 (0.83)	4.28 (2.12)	4.03 (2.24)
8	0.59 (0.34)	5.92 (1.78)	5.70 (1.74)

both the corrected FEM and the corrected network outputs in Table 4-5. Since we use a finer mesh, here, we skip the mesh refinement step in training. Due to limited data size, we use the same training and validation sequences for both the simulator and correction networks, which may reduce the performance of the correction network for the network predictions.

Table 4-5. Average (standard deviation) error in mm for each sequence between the observed point cloud (PC) and corrected meshes. Point cloud to mesh error is measured as average distance of every point on the point cloud to the closest point on the mesh. The improvement by using the correction network compared to Table 4-4 is given.

Seq.	Corrected FEM	Improvement	Corrected Network	Improvement
1	4.07 (2.52)	10.7%	4.13 (2.26)	7.8%
2	3.41 (1.40)	17.8%	3.39 (1.27)	15.3%
3	4.89 (0.56)	16.4%	4.85 (0.54)	13.9%
4	4.51 (1.41)	15.1%	4.44 (1.41)	12.1%
5	2.94 (1.34)	15.6%	2.67 (1.33)	16.3%
6	3.40 (1.53)	14.4%	3.22 (1.59)	17.0%
7	3.75 (2.02)	12.4%	3.64 (2.13)	9.7%
8	5.23 (1.69)	11.7%	5.46 (1.60)	4.2%

We observe that both sets of meshes also show similar performance after correction. This paves the path for a network-based soft-tissue simulator that can run at video-frame rates and can incorporate real-time observations.

4.4.3 Discussion

The proposed method is able to accurately simulate soft-tissue deformation by learning from FEM simulations. Its ability to run at video-frame rates at test time makes this approach also suitable for providing real-time feedback to users, such as for intraoperative guidance or a surgical simulator. After correction, its errors generally fall below the 5 mm identified as clinically useful [116]. As this is similar to the FEM error, improving the FEM by using a finer mesh may also improve the network predictions. Although the overall error is lower, we observe that the network predictions are smoother than that of the FEM. One potential solution for the over-smooth problem could be to train a GAN to sharpen the mesh, such as that used in [147].

In this work, we used fairly coarse meshes as we focused on having enough data with which to train a network. The size is similar to the cantilever used in [137]. An interesting extension is to include elements from networks used in plastic deformation such as in [140], which builds a graph in each step. This way, one could adopt the adaptive meshing used in FEMs [148] to better capture details in more deformed areas. The drawback is that the graph discovery step remains an open question for elastic materials. In plastic materials, the interactions only depend on each node's neighbors at each time step whereas in elastic materials, it also depends on a node's starting position.

Currently, all our training data sequences were selected to be at the same frequency. An extension to this work could incorporate time as a parameter of the network to account for simulations at different time steps. A next step can also look at generalizing to different geometries and material parameters. More work is needed to determine whether a network trained on one phantom could be fine-tuned to phantoms of different geometry and material.

Using real-time volumetric observations may improve data availability. Future work could use 3D ultrasound or other imaging sources to train directly from observations. This would skip the dependence on FEMs and entirely avoid the need to estimate material parameters and tissue boundaries. As collecting 3D observations of deformation is time-consuming, e.g. by using 3D ultrasound, or may require high radiation exposure, e.g. by using CT scans, we are also considering options to augment limited observations. One potential augmentation is methods that estimate the entire mesh from partial observation, such as [110], thereby replacing training from FEM with observations. Combining this with methods that construct point clouds from endoscope videos, such as [117], could allow data from any robotic procedure on a phantom with known geometry to be a potential source of training data.

To use the techniques proposed in this chapter on real organs, one would start with segmenting the organ in question from a preoperative scan. This would give us a patient specific organ shape from which we can construct a volumetric mesh and simulate using an FEM. The segmentation may need to be finer than organ-level to distinguish between different types of tissues that may have different properties. For the material parameters, we could start with a best guess for the stiffness of the different parts of organs. This would be calculated from the average value across patients, modified by any pre-existing conditions (such as fatty liver disease increasing stiffness). We can then train a network to mimic the FEM results as shown in Section 4.4. The network provides soft-tissue simulation that can track the deformation of the patient in real time, given the robot instrument poses. We may need to develop segmentation algorithms to distinguish between the different parts of the organs. Future work could add forces, calculated from the method proposed in Chapter 2, as an input to the simulator.

The next step is to train one or more correction network as the material parameters may be imprecise and modeling boundary conditions in soft tissue is not well understood. Each different type of tissue could be a different graph where internal edges within each graph learn to correct for material parameters. We can construct edges between graphs to form the boundary conditions between the different tissues. This requires observing interactions with the patient and may require additional imaging modalities. While we used only surface observations in this chapter, observations would ideally be volumetric in order to ensure the entire volume's deformation is accurate. One way to do this is to use a tracked ultrasound probe and register the ultrasound image to the pre-operative scan such as through the method outlined by Kapoor et al. [149]. We could compare the ultrasound scan to how the trained network has estimated the organ to have deformed and correct for discrepancies. Further studies on how much data is required to train the correction network, similar to the analysis done in Section 2.5.3.1, is needed to see when the correction network is sufficiently trained.

4.5 Conclusion

In this chapter, we outline a framework to correct model-based simulation using data readily available in robotic surgeries. We also outline a first step of a deep-learning based soft-tissue simulator that can provide step-wise estimations of deformations that mimic model-based simulators at real-time speeds. These methods have been integrated with the dVRK environment and can be incorporated in the open-source libraries as an interactive, accurate soft-tissue simulator. Additionally, an open source Blender client is provided to facilitate visualization of deforming meshes. These contributions can be used to provide real-time simulations and would enable future research that requires interacting with phantoms in simulated settings, such as training tasks and surgical challenges.

We validate our correction network by implementing a network that predicts deformation corrections on a soft-tissue phantom during robot interactions. This network can correct for inaccurately modeled parameters as we show improvements across a wide range of Young's Modulus. Other terms like friction and boundary conditions are rarely available in patient data so any model-based simulation would have an irreducible error. A data-driven correction factor could account for difficult-to-model errors and a network-based simulator can run in real time to provide intraoperative guidance. We envision that the networks can be used together to provide guidance for clinicians. The network-based simulator can enable intraoperative use, and the correction network can be adapted for life-long learning throughout a surgery as the patient condition changes.

Chapter 5

Conclusions and future outlook

This dissertation presents ways to use machine learning to build understanding of the surgical environment. Its contributions are to explore using self-supervised learning to build representations in three areas: robot signals, surgeon intent, and patient scene dynamics. The main focus is to use observable measurements as a proxy to learn unobservable underlying states.

Chapter 1 introduces the motivation to build intelligent robotic surgical systems. It presents some cases where robots are already used in the operating room at comparatively high levels of autonomy and discusses how an intelligent surgical assistant could help surgeons. In contrast to highly structured environments such as orthopedics, the challenge in autonomy in laparoscopic procedures is in understanding how the anatomy aligns with the pre-operative scans. Since the anatomy is much more deformable, jumping directly to high-levels of autonomy is technically challenging. Instead, a pathway could involve where the robot supports the surgeon's actions while the surgeon is still in full control. To achieve this, we need to understand not only the surgical plan but also the surgeon's current actions and the robot state. The next three chapters look at building understanding of each of these components.

Chapter 2 presents two ways to learn a model for force sensing: either explicitly through learning the forces directly or implicitly through learning free space torques.

The direct method does not require the Jacobian of the robot system, but it is limited in its ability to generalize to different areas of the workspace. The indirect method is more generalizable and does not require a sensor to train. This lends itself well to being deployed clinically, where it can learn a correction step for surgery-specific interactions. We demonstrated that the learned force sensing is accurate within 2 N and could benefit inexperienced users for palpation tasks.

Chapter 3 uses cross-modal supervision to learn representations of surgical actions from video and kinematics. These representations cluster based on skill and gesture without having seen either label in training. We demonstrate that classifiers trained on these representations can identify separations in gesture ($64 - 68\%$ accuracy) and skill ($63 - 81\%$ accuracy) classes. Furthermore, the representation of novel tasks not seen at training produces representations that fall mainly within the novice class. This could be used to identify new gestures if we observe representations far from where we expect them to fall for a specific surgeon. We see this representation as a step towards learning personalized predictions for each surgeon. Being able to identify new gestures could lead to life long learning that improves over examples from different tasks.

Chapter 4 presents learning-based methods to learn soft-tissue deformations from intraoperative observations and from models. It presents two models: one for correcting soft-tissue simulation results and one for mimicking physics-based simulator results at real-time speeds using a neural network. These could provide guidance to surgeons for structures that are below organ surfaces through augmented reality. Accurate real time simulations, in combination with an understanding of surgeon actions and robot states, could also form a basis for the robot to take automated actions to help surgeons achieve their goals. Given a close enough model of the soft-tissue, we added a depth camera to measure how a phantom actually deforms vs. how the simulation estimated that it would deform. We train a neural network to correct the simulation. It is self-supervised and can learn throughout a surgery. As material parameters and boundary conditions are seldom accurate for patients and cannot be updated as patient conditions change, it is crucial to add a correction based on observations.

While there is increasing interest in adding more intelligence into robotic systems, a general and holistic understanding of surgery remains a technical challenge. By using self-supervision, the methods proposed in this thesis shift away from narrowly focused methods to solve one particular problem. Instead, they aim to construct a more holistic understanding of the scene that can update throughout a surgery. As labeling data scales poorly and is difficult to personalize, fully-supervised approaches necessarily preclude personalized predictions. In contrast, self-supervised and unsupervised learning can be trained throughout and across surgeries.

With more generalized understanding of surgeries, I hope to build systems that can reason about all the components together. Understanding the surgeon's intent should be predicated upon what the patient anatomy looks like in the scene. Transitions between surgical gestures (e.g., the start and stop points for suturing) have been hard to estimate and I believe part of the reason is due to the lack of understanding of the scene. Existing learning algorithms may recognize that the task is suturing, and while computer vision algorithms may be able to identify where the opening is, few works can use both pieces of information to make clinical decisions about where to place each suture. In order for robots to become intelligent assistants in the operating room, we must be able to reason over all the pieces of surgery in a unified framework.

5.1 Future work

While the representations in this thesis are an avenue to life-long learning, they still are limited in the information from which they are learning. In order for the robot to perform autonomous actions, it must understand each of the components in context of each other. An interesting next step would be to learn representations that are informed by more than one of the components of a robot-assisted surgery (RAS), such as the surgeon action in the context of the patient scene or how interacting with different parts of the anatomy affects the robot state.

The representations proposed by this thesis are all learned from robot systems designed for human tele-operation. As such, these systems' feedback are mainly meant to be human interpretable. Chapter 4 and one method presented in Chapter 2 make use of sensor and camera readings not typically available during RAS. Yet, to fully realize the benefits of robotic systems, future systems should include more sensors. Examples could be more cameras that go beyond stereo scene reconstruction, or infrared cameras to do intraoperative tissue characterization to update the soft-tissue model.

While I have used surgeon actions as the gold standard in this dissertation, they may not be the optimal steps for every surgery. For example, LASIK follows eye movements in ways that a surgeon could not. In this case, the representations learned in Chapter 3 would become irrelevant, though alternative metrics like surgery and recovery time would be useful to determine an optimal path. The representations presented in Chapter 2 and Chapter 4 become more relevant as the robot must be able to perform completely autonomously according to its modeling of its own state and the scene. This may even improve accuracy of procedures. While surgeons depend on experience to learn how tissue deforms, robots could learn this more quantitatively. It could palpate the tissue and use the force estimation to update the material parameters and boundary conditions for the simulator to improve its scene modeling.

To go a step further, future innovations in robotics may bring about different capabilities that have no obvious mapping to human actions. Ideal surgical robots would require no incision on the outside. They may take the form of nano robots that the patient swallows as a pill and then can travel through the body based on autonomous path-planning algorithms to perform precise cuts from within. There, it becomes more important that the robot can assess the accuracy of its model of the scene and whether it has accomplished its goal. Research that studies how the robot can communicate its intent to the surgeon would be useful to double check that the robot's model of the scene is accurate and that its plan to carry out the surgery is reasonable.

One of the main challenges in adopting intelligent assistants will be to bound errors in clinical settings. Right now, artificial intelligence systems are not necessarily designed to be interpretable. While there are some works to address this and introduce causal inference in medical image understanding [150], surgery introduces additional challenges in time constraints. We need algorithms that can measure uncertainty and more human-robot interaction studies for how to convey uncertainty to surgeons. We should also study how robots might automate certain subtasks or drive additional arms while humans maintain control of the surgery. One example of where the robot can assist the surgeon in shared autonomy schemes could be in controlling the endoscope movement and a suctioning tool to keep the field of view clear. To keep the surgeon in control, we need ways for the robot to signal intent to the surgeon before it takes an action, such as moving the endoscope. This would create a partnership between the surgeon and the robot that elevates the surgical systems beyond a simple extension of the surgeon's arms.

References

1. Velanovich, V. Laparoscopic vs open surgery. *Surgical endoscopy* **14,** 16–21 (2000).
2. Agha, R. & Muir, G. Does laparoscopic surgery spell the end of the open surgeon? *Journal of the Royal Society of Medicine* **96,** 544–546 (2003).
3. Awasthi, R. C. A Comparative Study to Evaluate the Laparoscopic vs Open Surgery Cholecystectomy: A Prospective Study. *Journal of Advanced Medical and Dental Sciences Research* **6,** 94–97 (2018).
4. Sheetz, K. H., Claflin, J. & Dimick, J. B. Trends in the adoption of robotic surgery for common surgical procedures. *JAMA network open* **3,** e1918911–e1918911 (2020).
5. Dao, T. M. H., Nguyen, T. K. T. & Schirling, D. *The future of robotic-assisted laparoscopic surgery* (Hochschule Furtwangen, 2020).
6. Birkmeyer, J. D., Finks, J. F., O'Reilly, A., Oerline, M., Carlin, A. M., Nunn, A. R., Dimick, J., Banerjee, M. & Birkmeyer, N. J. Surgical skill and complication rates after bariatric surgery. *New England Journal of Medicine* **369,** 1434–1442 (2013).
7. Heemskerk, J., van Gemert, W. G., de Vries, J., Greve, J. & Bouvy, N. D. Learning curves of robot-assisted laparoscopic surgery compared with conventional laparoscopic surgery: an experimental study evaluating skill acquisition of robot-assisted laparoscopic tasks compared with conventional laparoscopic tasks in inexperienced users. *Surgical Laparoscopy Endoscopy & Percutaneous Techniques* **17,** 171–174 (2007).
8. Chen, I.-H. A., Ghazi, A., Sridhar, A., Stoyanov, D., Slack, M., Kelly, J. D. & Collins, J. W. Evolving robotic surgery training and improving patient safety, with the integration of novel technologies. *World Journal of Urology,* 1–11 (2020).
9. Sridhar, A. N., Briggs, T. P., Kelly, J. D. & Nathan, S. Training in robotic surgery—an overview. *Current urology reports* **18,** 1–8 (2017).
10. Jenison, E. L., Gil, K. M., Lendvay, T. S. & Guy, M. S. Robotic surgical skills: acquisition, maintenance, and degradation. *JSLS: Journal of the Society of Laparoendoscopic Surgeons* **16,** 218 (2012).
11. Birkmeyer, J. D. The Future of Artificial Intelligence in Surgical Care. *Annals of Surgery* **272,** 529 (2020).
12. Guru, K. A., Esfahani, E. T., Raza, S. J., Bhat, R., Wang, K., Hammond, Y., Wilding, G., Peabody, J. O. & Chowriappa, A. J. Cognitive skills assessment during robot-assisted surgery: separating the wheat from the chaff. *BJU international* **115,** 166–174 (2015).

13. Yang, G.-Z., Cambias, J., Cleary, K., Daimler, E., Drake, J., Dupont, P. E., Hata, N., Kazanzides, P., Martel, S., Patel, R. V., Santos, V. J. & Taylor, R. H. Medical robotics—Regulatory, ethical, and legal considerations for increasing levels of autonomy. *Science Robotics* **2,** 8638 (2017).

14. Bargar, W., DiGioia, A., Turner, R., Taylor, J., McCarthy, J. & Mears, D. *Robodoc Multi-Center Trial: An Interim Report* in *Proc. 2nd Int. Symp. on Medical Robotics and Computer Assisted Surgery* (1995), 208–214.

15. Taylor, R. H., Mittelstadt, B. D., Paul, H. A., Hanson, W., Kazanzides, P., Zuhars, J. F., Williamson, B., Musits, B. L., Glassman, E. & Bargar, W. L. An image-directed robotic system for precise orthopaedic surgery. *IEEE Transactions on Robotics and Automation* **10,** 261–275 (1994).

16. Paul, H., Mittelstadt, B., Bargar, W., Kazanzides, P. & Williamson, B. *Accuracy of Implant Interface Preparation: Hand-held Broach vs. Robot Machine Tool* in *Proc. Orthopaedic Research Society* (1992).

17. Paul, H. A., Bargar, W. L., Mittlestadt, B., Musits, B., Taylor, R. H., Kazanzides, P., Zuhars, J., Williamson, B. & Hanson, W. Development of a surgical robot for cementless total hip arthroplasty. *Clinical Orthopaedics and related research,* 57–66 (1992).

18. Kazanzides, P., Mittelstadt, B. D., Musits, B. L., Bargar, W. L., Zuhars, J. F., Williamson, B., Cain, P. W. & Carbone, E. J. An integrated system for cementless hip replacement. *IEEE Engineering in Medicine and Biology Magazine* **14,** 307–313 (1995).

19. Adler Jr, J. R., Chang, S. D., Murphy, M. J., Doty, J., Geis, P. & Hancock, S. L. The Cyberknife: a frameless robotic system for radiosurgery. *Stereotactic and functional neurosurgery* **69,** 124–128 (1997).

20. Shademan, A., Decker, R. S., Opfermann, J. D., Leonard, S., Krieger, A. & Kim, P. C. Supervised autonomous robotic soft tissue surgery. *Science translational medicine* **8,** 337ra64–337ra64 (2016).

21. Li, Y., Richter, F., Lu, J., Funk, E. K., Orosco, R. K., Zhu, J. & Yip, M. C. Super: A surgical perception framework for endoscopic tissue manipulation with surgical robotics. *Robotics and Automation Letters* **5,** 2294–2301 (2020).

22. Murali, A., Sen, S., Kehoe, B., Garg, A., McFarland, S., Patil, S., Boyd, W. D., Lim, S., Abbeel, P. & Goldberg, K. *Learning by observation for surgical subtasks: Multilateral cutting of 3d viscoelastic and 2d orthotropic tissue phantoms* in *Int. Conf. on Robotics and Automation (ICRA)* (2015), 1202–1209.

23. Korbar, B., Tran, D. & Torresani, L. *Cooperative learning of audio and video models from self-supervised synchronization* in *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (2018), 7774–7785.

24. Lu, J., Jayakumari, A., Richter, F., Li, Y. & Yip, M. C. Super deep: A surgical perception framework for robotic tissue manipulation using deep learning for feature extraction. *arXiv preprint arXiv:2003.03472* (2020).

25. Haouchine, N., Dequidt, J., Peterlik, I., Kerrien, E., Berger, M.-O. & Cotin, S. *Image-guided simulation of heterogeneous tissue deformation for augmented reality during hepatic surgery* in *International symposium on mixed and augmented reality (ISMAR)* (2013), 199–208.

26. DiPietro, R., Lea, C., Malpani, A., Ahmidi, N., Vedula, S. S., Lee, G. I., Lee, M. R. & Hager, G. D. *Recognizing surgical activities with recurrent neural networks* in *International conference on medical image computing and computer-assisted intervention* (2016), 551–558.

27. DiPietro, R. & Hager, G. D. *Unsupervised learning for surgical motion by learning to predict the future* in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (2018), 281–288.

28. Kazanzides, P., Chen, Z., Deguet, A., Fischer, G. S., Taylor, R. H. & DiMaio, S. P. *An open-source research kit for the da Vinci® Surgical System* in *Int. Conf. on Robotics and Automation (ICRA)* (2014), 6434–6439.

29. Okamura, A. M. Methods for haptic feedback in teleoperated robot-assisted surgery. *Industrial Robot: An International Journal* (2004).

30. Hannaford, B., Rosen, J., Friedman, D. W., King, H., Roan, P., Cheng, L., Glozman, D., Ma, J., Kosari, S. N. & White, L. Raven-II: an open platform for surgical robotics research. *IEEE Transactions on Biomedical Engineering* **60,** 954–959 (2012).

31. Tran, N., Wu, J. Y., Deguet, A. & Kazanzides, P. *A Deep Learning Approach to Intrinsic Force Sensing on the da Vinci Surgical Robot* in *Int. Conf. on Robotic Computing (IRC)* (2020), 25–32.

32. Yilmaz, N., Wu, J. Y., Kazanzides, P. & Tumerdem, U. *Neural network based inverse dynamics identification and external force estimation on the da Vinci Research Kit* in *Int. Conf. on Robotics and Automation (ICRA)* (2020), 1387–1393.

33. Wu, J. Y., Yilmaz, N., Tumerdem, U. & Kazanzides, P. *Robot Force Estimation with Learned Intraoperative Correction* in *International Symposium on Medical Robotics (ISMR)* (2021).

34. Puangmali, P., Liu, H., Seneviratne, L. D., Dasgupta, P. & Althoefer, K. Miniature 3-axis distal force sensor for minimally invasive surgical palpation. *Ieee/Asme Transactions On Mechatronics* **17,** 646–656 (2011).

35. Faragasso, A., Bimbo, J., Noh, Y., Jiang, A., Sareh, S., Liu, H., Nanayakkara, T., Wurdemann, H. A. & Althoefer, K. *Novel uniaxial force sensor based on visual information for minimally invasive surgery* in *Int. Conf. on Robotics and Automation (ICRA)* (2014), 1405–1410.

36. Yip, M. C., Yuen, S. G. & Howe, R. D. A robust uniaxial force sensor for minimally invasive surgery. *IEEE transactions on biomedical engineering* **57,** 1008–1011 (2010).

37. Puangmali, P., Althoefer, K., Seneviratne, L. D., Murphy, D. & Dasgupta, P. State-of-the-art in force and tactile sensing for minimally invasive surgery. *IEEE Sensors Journal* **8,** 371–381 (2008).

38. Hagn, U., Konietschke, R., Tobergte, A., Nickl, M., Jörg, S., Kübler, B., Passig, G., Gröger, M., Fröhlich, F., Seibold, U., Le-Tien, L., Albu-Schäffer, A., Nothhelfer, A., Hacker, F., Grebenstein, M. & Hirzinger, G. DLR MiroSurge: a versatile system for research in endoscopic telesurgery. *International journal of computer assisted radiology and surgery* **5,** 183–193 (2010).

39. Berkelman, P. J., Whitcomb, L. L., Taylor, R. H. & Jensen, P. A miniature microsurgical instrument tip force sensor for enhanced force feedback during robot-assisted manipulation. *IEEE Transactions on Robotics and Automation* **19,** 917–921 (2003).

40. Seibold, U., Kubler, B. & Hirzinger, G. *Prototype of instrument for minimally invasive surgery with 6-axis force sensing capability* in *IEEE International Conference on Robotics and Automation* (2005), 496–501.

41. Kim, U., Lee, D.-H., Yoon, W. J., Hannaford, B. & Choi, H. R. Force sensor integrated surgical forceps for minimally invasive robotic surgery. *IEEE Transactions on Robotics* **31,** 1214–1224 (2015).

42. Peña, R., Smith, M. J., Ontiveros, N. P., Hammond, F. L. & Wood, R. J. *Printing strain gauges on Intuitive Surgical da Vinci robot end effectors* in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2018), 806–812.

43. Peirs, J., Clijnen, J., Reynaerts, D., Van Brussel, H., Herijgers, P., Corteville, B. & Boone, S. A micro optical force sensor for force feedback during minimally invasive robotic surgery. *Sensors and Actuators A: Physical* **115,** 447–455 (2004).

44. Shahzada, K. S., Yurkewich, A., Xu, R. & Patel, R. V. *Sensorization of a surgical robotic instrument for force sensing* in *Optical Fibers and Sensors for Medical Diagnostics and Treatment Applications XVI* **9702** (2016), 97020U.

45. Willaert, B., Famaey, N., Verbrugghe, P., Reynaerts, D. & Van Brussel, H. *Design and in vivo validation of a force-measuring manipulator for MIS providing synchronized video, motion and force data* in *IEEE International Conference on Robotics and Automation* (2013), 4857–4862.

46. Shimachi, S., Hakozaki, Y., Tada, T. & Fujiwara, Y. *Measurement of force acting on surgical instrument for force-feedback to master robot console* in *International Congress Series* **1256** (2003), 538–546.

47. Schwalb, W., Shirinzadeh, B. & Smith, J. A force-sensing surgical tool with a proximally located force/torque sensor. *The International Journal of Medical Robotics and Computer Assisted Surgery* **13,** e1737 (2017).

48. Fontanelli, G. A., Buonocore, L. R., Ficuciello, F., Villani, L. & Siciliano, B. *A novel force sensing integrated into the trocar for minimally invasive robotic surgery* in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2017), 131–136.

49. O'Neill, J. J., Stephens, T. K. & Kowalewski, T. M. Evaluation of torque measurement surrogates as applied to grip torque and jaw angle estimation of robotic surgical tools. *Robotics and Automation Letters* **3,** 3027–3034 (2018).

50. Guo, Y., Pan, B., Fu, Y. & Meng, M. Q.-H. *Grip Force Perception Based on dAENN for Minimally Invasive Surgery Robot* in *IEEE International Conference on Robotics and Biomimetics (ROBIO)* (2019), 1216–1221.

51. Yu, L., Yu, X. & Zhang, Y. Microinstrument contact force sensing based on cable tension using BLSTM–MLP network. *Intelligent Service Robotics* **13,** 123–135 (2020).

52. Gao, C., Liu, X., Peven, M., Unberath, M. & Reiter, A. in *OR 2.0 Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis* 118–127 (Springer, 2018).

53. Aviles, A. I., Marban, A., Sobrevilla, P., Fernandez, J. & Casals, A. *A recurrent neural network approach for 3d vision-based force estimation* in *2014 4th International Conference on Image Processing Theory, Tools and Applications (IPTA)* (2014), 1–6.

54. Aviles, A. I., Alsaleh, S. M., Sobrevilla, P. & Casals, A. *Force-feedback sensory substitution using supervised recurrent learning for robotic-assisted surgery* in *37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (2015), 1–4.

55. Aviles, A. I., Alsaleh, S. M., Hahn, J. K. & Casals, A. Towards retrieving force feedback in robotic-assisted surgery: A supervised neuro-recurrent-vision approach. *IEEE Transactions on Haptics* **10,** 431–443 (2016).

56. Gessert, N., Beringhoff, J., Otte, C. & Schlaefer, A. Force estimation from OCT volumes using 3D CNNs. *International journal of computer assisted radiology and surgery* **13,** 1073–1082 (2018).

57. Noohi, E., Parastegari, S. & Žefran, M. *Using monocular images to estimate interaction forces during minimally invasive surgery* in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2014), 4297–4302.

58. Marban, A., Srinivasan, V., Samek, W., Fernández, J. & Casals, A. *Estimation of interaction forces in robotic surgery using a semi-supervised deep neural network model* in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (2018), 761–768.

59. Madhani, A. J., Niemeyer, G. & Salisbury, J. K. *The black falcon: a teleoperated surgical instrument for minimally invasive surgery* in *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No. 98CH36190)* **2** (1998), 936–944.

60. Mahvash, M., Gwilliam, J., Agarwal, R., Vagvolgyi, B., Su, L.-M., Yuh, D. D. & Okamura, A. M. *Force-feedback surgical teleoperator: Controller design and palpation experiments* in *2008 Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems* (2008), 465–471.

61. Haghighipanah, M., Miyasaka, M. & Hannaford, B. Utilizing elasticity of cable-driven surgical robot to estimate cable tension and external force. *Robotics and Automation Letters* **2,** 1593–1600 (2017).

62. Lee, M. C., Kim, C. Y., Yao, B., Peine, W. J. & Song, Y. E. *Reaction force estimation of surgical robot instrument using perturbation observer with SMCSPO algorithm* in *2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics* (2010), 181–186.

63. Tadano, K. & Kawashima, K. *Development of 4-DOFs forceps with force sensing using pneumatic servo system* in *Int. Conf. on Robotics and Automation* (2006), 2250–2255.

64. Takahashi, H., Warisawa, S., Mitsuishi, M., Arata, J. & Hashizume, M. *Development of high dexterity minimally invasive surgical system with augmented force feedback capability* in *IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics* (2006), 284–289.

65. Seneci, C. A., Leibrandt, K., Wisanuvej, P., Shang, J., Darzi, A. & Yang, G.-Z. *Design of a smart 3D-printed wristed robotic surgical instrument with embedded force sensing and modularity* in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2016), 3677–3683.

66. Yilmaz, N., Bazman, M. & Tumerdem, U. *External force/torque estimation on a dexterous parallel robotic surgical instrument wrist* in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2018), 4396–4403.

67. Yilmaz, N., Bazman, M., Alassi, A., Gur, B. & Tumerdem, U. *6-Axis Hybrid Sensing and Estimation of Tip Forces/Torques on a Hyper-Redundant Robotic Surgical Instrument* in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2019), 2990–2997.

68. Sang, H., Yun, J., Monfaredi, R., Wilson, E., Fooladi, H. & Cleary, K. External force estimation and implementation in robotically assisted minimally invasive surgery. *The International Journal of Medical Robotics and Computer Assisted Surgery* **13,** e1824 (2017).

69. Fontanelli, G. A., Ficuciello, F., Villani, L. & Siciliano, B. *Modelling and identification of the da Vinci research kit robotic arms* in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2017), 1464–1469.

70. Piqué, F., Boushaki, M. N., Brancadoro, M., De Momi, E. & Menciassi, A. *Dynamic modeling of the da Vinci Research Kit arm for the estimation of interaction wrench* in *International Symposium on Medical Robotics (ISMR)* (2019), 1–7.

71. Wang, Y., Gondokaryono, R., Munawar, A. & Fischer, G. S. A convex optimization-based dynamic model identification package for the da Vinci Research Kit. *Robotics and Automation Letters* **4,** 3657–3664 (2019).

72. Colomé, A., Pardo, D., Alenya, G. & Torras, C. *External force estimation during compliant robot manipulation* in *Int. Conf. on Robotics and Automation (ICRA)* (2013), 3535–3540.

73. Mozaffari, A., Behzadipour, S. & Kohani, M. Identifying the tool-tissue force in robotic laparoscopic surgery using neuro-evolutionary fuzzy systems and a synchronous self-learning hyper level supervisor. *Applied Soft Computing* **14,** 12–30 (2014).

74. Yasin, R. & Simaan, N. Joint-level force sensing for indirect hybrid force/position control of continuum robots with friction. *The International Journal of Robotics Research* **40,** 764–781 (2021).

75. Smith, A. C., Mobasser, F. & Hashtrudi-Zaad, K. Neural-network-based contact force observers for haptic applications. *IEEE Transactions on Robotics* **22,** 1163–1175 (2006).

76. Abeywardena, S., Yuan, Q., Tzemanaki, A., Psomopoulou, E., Droukas, L., Melhuish, C. & Dogramadzi, S. Estimation of tool-tissue forces in robot-assisted minimally invasive surgery using neural networks. *Frontiers in Robotics and AI* **6,** 56 (2019).

77. Wu, J. Y., Chen, Z., Deguet, A. & Kazanzides, P. *FPGA-based velocity estimation for control of robots with low-resolution encoders* in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (2018), 6384–6389.

78. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R. & Ng, A. Y. *ROS: an open-source Robot Operating System* in *ICRA Workshop on Open Source Software* (2009).

79. Lin, H., Hui, C.-W. V., Wang, Y., Deguet, A., Kazanzides, P. & Au, K. S. A reliable gravity compensation control strategy for dvrk robotic arms with nonlinear disturbance forces. *Robotics and Automation Letters* **4,** 3892–3899 (2019).

80. Chollet, F. *et al. Keras* https://github.com/fchollet/keras. 2015.

81. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Y., Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu & Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* Software available from tensorflow.org. 2015.

82. MacKay, D. J. Bayesian interpolation. *Neural computation* **4,** 415–447 (1992).

83. Widrow, B. & Lehr, M. A. Adaptive neural networks and their applications. *International Journal of Intelligent Systems* **8,** 453–507 (1993).

84. Hwang, M., Thananjeyan, B., Paradis, S., Seita, D., Ichnowski, J., Fer, D., Low, T. & Goldberg, K. Efficiently Calibrating Cable-Driven Surgical Robots With RGBD Sensing, Temporal Windowing, and Linear and Recurrent Neural Network Compensation. *arXiv preprint arXiv:2003.08520* (2020).

85. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. & Chintala, S. in *Advances in Neural Information Processing Systems 32* (eds Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. & Garnett, R.) 8024–8035 (Curran Associates, Inc., 2019).

86. Chalasani, P., Wang, L., Yasin, R., Simaan, N. & Taylor, R. H. Preliminary Evaluation of an Online Estimation Method for Organ Geometry and Tissue Stiffness. *IEEE Robotics and Automation Letters* **3,** 1816–1823 (2018).

87. Guthart, G. S. & Salisbury, J. K. *The Intuitive^{TM} telesurgery system: overview and application* in *Int. Conf. on Robotics and Automation* **1** (2000), 618–621.

88. Sarikaya, D. & Jannin, P. Surgical Gesture Recognition with Optical Flow only. *arXiv preprint arXiv:1904.01143* (2019).

89. DiPietro, R. & Hager, G. D. *Automated surgical activity recognition with one labeled sequence* in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (2019), 458–466.

90. Long, Y.-H., Wu, J. Y., Lu, B., Jin, Y.-M., Unberath, M., Liu, Y.-H., Heng, P.-A. & Dou, Q. *Relational Graph Learning on Visual and Kinematics Embeddings for Accurate Gesture Recognition in Robotic Surgery* in *Int. Conf. on Robotics and Automation* (2021).

91. Qin, Y., Feyzabadi, S., Allan, M., Burdick, J. W. & Azizian, M. daVinciNet: Joint Prediction of Motion and Surgical State in Robot-Assisted Surgery. *arXiv preprint arXiv:2009.11937* (2020).

92. Wu, J. Y., Tamhane, A., Kazanzides, P. & Unberath, M. Cross-modal self-supervised representation learning for gesture and skill recognition in robotic surgery. *International Journal of Computer Assisted Radiology and Surgery* **16,** 779–787 (2021).

93. Jing, L. & Tian, Y. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

94. Arandjelovic, R. & Zisserman, A. *Objects that sound* in *Proceedings of the European Conference on Computer Vision* (2018), 435–451.

95. Zhang, Y. & Lu, H. *Deep cross-modal projection learning for image-text matching* in *Proceedings of the European Conference on Computer Vision* (2018), 686–701.

96. Zhen, L., Hu, P., Wang, X. & Peng, D. *Deep supervised cross-modal retrieval* in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition* (2019), 10394–10403.

97. Mazomenos, E., Watson, D., Kotorov, R. & Stoyanov, D. *Gesture Classification in Robotic Surgery using Recurrent Neural Networks with Kinematic Information* in *8th Joint Workshop on New Technologies for Computer/Robotic Assisted Surgery* (Sept. 2018).

98. Ahmidi, N., Tao, L., Sefati, S., Gao, Y., Lea, C., Haro, B. B., Zappella, L., Khudanpur, S., Vidal, R. & Hager, G. D. A dataset and benchmarks for segmentation and recognition of gestures in robotic surgery. *Transactions on Biomedical Engineering* **64,** 2025–2041 (2017).

99. Funke, I., Mees, S. T., Weitz, J. & Speidel, S. Video-based surgical skill assessment using 3D convolutional neural networks. *International journal of computer assisted radiology and surgery* **14,** 1217–1225 (2019).

100. Tanwani, A. K., Sermanet, P., Yan, A., Anand, R., Phielipp, M. & Goldberg, K. Motion2Vec: Semi-Supervised Representation Learning from Surgical Videos. *arXiv preprint arXiv:2006.00545* (2020).

101. Murali, A., Garg, A., Krishnan, S., Pokorny, F. T., Abbeel, P., Darrell, T. & Goldberg, K. *Tsc-dl: Unsupervised trajectory segmentation of multi-modal surgical demonstrations with deep learning* in *IEEE Int. Conf. on Robotics and Automation* (2016), 4150–4157.

102. Van Amsterdam, B., Nakawala, H., De Momi, E. & Stoyanov, D. *Weakly supervised recognition of surgical gestures* in *IEEE Int. Conf. on Robotics and Automation* (2019), 9565–9571.

103. Gao, Y., Vedula, S. S., Reiley, C. E., Ahmidi, N., Varadarajan, B., Lin, H. C., Tao, L., Zappella, L., Béjar, B., Yuh, D. D., Chen, C. C. G., Vidal, R., Khudanpur, S. & Hager, G. D. *JHU-ISI gesture and skill assessment working set (jigsaws): A surgical activity dataset for human motion modeling* in *MICCAI Workshop: M2CAI* **3** (2014), 3.

104. Farnebäck, G. *Two-frame motion estimation based on polynomial expansion* in *Scandinavian Conference on Image Analysis* (2003), 363–370.

105. Simonyan, K. & Zisserman, A. *Two-stream convolutional networks for action recognition in videos* in *Advances in Neural Information Processing Systems* (2014), 568–576.

106. Chen, T. & Guestrin, C. *Xgboost: A scalable tree boosting system* in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), 785–794.

107. McInnes, L., Healy, J. & Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018).

108. Wang, Z. & Fey, A. M. Deep learning with convolutional neural network for objective skill evaluation in robot-assisted surgery. *International journal of computer assisted radiology and surgery* **13,** 1959–1970 (2018).

109. Mohareri, O., Ischia, J., Black, P. C., Schneider, C., Lobo, J., Goldenberg, L. & Salcudean, S. E. Intraoperative registered transrectal ultrasound guidance for robot-assisted laparoscopic radical prostatectomy. *The Journal of Urology* **193,** 302–312 (2015).

110. Pfeiffer, M., Riediger, C., Weitz, J. & Speidel, S. Learning soft tissue behavior of organs for surgical navigation with convolutional neural networks. *International journal of computer assisted radiology and surgery* **14,** 1147–1155 (2019).

111. Suwelack, S., Röhl, S., Bodenstedt, S., Reichard, D., Dillmann, R., dos Santos, T., Maier-Hein, L., Wagner, M., Wünscher, J., Kenngott, H., Müller, B. P. & Speidel, S. Physics-based shape matching for intraoperative image guidance. *Medical Physics* **41,** 111901 (2014).

112. Cotin, S., Delingette, H. & Ayache, N. Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions on Visualization and Computer Graphics* **5,** 62–73 (1999).

113. Fialko, S. & Karpilowskyi, V. *Multithreaded parallelization of the finite element method algorithms for solving physically nonlinear problems* in *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)* (2018), 311–318.

114. Courtecuisse, H., Allard, J., Kerfriden, P., Bordas, S. P., Cotin, S. & Duriez, C. Real-time simulation of contact and cutting of heterogeneous soft-tissues. *Medical Image Analysis* **18,** 394–410 (2014).

115. Zhang, J., Zhong, Y., Smith, J. & Gu, C. A new ChainMail approach for real-time soft tissue simulation. *Bioengineered* **7,** 246–252 (2016).

116. Brunet, J.-N., Mendizabal, A., Petit, A., Golse, N., Vibert, E. & Cotin, S. *Physics-based deep neural network for augmented reality during liver surgery* in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (2019), 137–145.

117. Liu, X., Sinha, A., Ishii, M., Hager, G. D., Reiter, A., Taylor, R. H. & Unberath, M. Dense Depth Estimation in Monocular Endoscopy with Self-supervised Learning Methods. *Transactions on medical imaging* (2019).

118. Wu, J. Y., Kazanzides, P. & Unberath, M. Leveraging vision and kinematics data to improve realism of biomechanic soft tissue simulation for robotic surgery. *International journal of computer assisted radiology and surgery* **15,** 811–818 (2020).

119. Shin, C., Ferguson, P. W., Pedram, S. A., Ma, J., Dutson, E. P. & Rosen, J. *Autonomous tissue manipulation via surgical robot using learning based model predictive control* in *2019 International Conference on Robotics and Automation (ICRA)* (2019), 3875–3881.

120. Richter, F., Orosco, R. K. & Yip, M. C. Open-sourced reinforcement learning environments for surgical robotics. *arXiv preprint arXiv:1903.02090* (2019).

121. Wu, J. Y., Munawar, A., Unberath, M. & Kazanzides, P. *Learning Soft-Tissue Simulation from Models and Observation* in *International Symposium on Medical Robotics (ISMR)* (2021).

122. Joldes, G., Bourantas, G., Zwick, B., Chowdhury, H., Wittek, A., Agrawal, S., Mountris, K., Hyde, D., Warfield, S. K. & Miller, K. Suite of meshless algorithms for accurate computation of soft tissue deformation for surgical simulation. *Medical Image Analysis* **56,** 152–171 (2019).

123. Zhang, J., Zhong, Y. & Gu, C. Neural network modelling of soft tissue deformation for surgical simulation. *Artificial Intelligence in Medicine* **97,** 61–70 (2019).

124. San-Vicente, G., Aguinaga, I. & Celigueta, J. T. Cubical mass-spring model design based on a tensile deformation test and nonlinear material model. *IEEE Transactions on Visualization and Computer Graphics* **18,** 228–241 (2011).

125. Zhang, J., Zhong, Y. & Gu, C. Deformable models for surgical simulation: a survey. *Reviews in Biomedical Engineering* **11,** 143–164 (2017).

126. Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T. & Ronneberger, O. *3D U-Net: learning dense volumetric segmentation from sparse annotation* in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (2016), 424–432.

127. Gondokaryono, R. A., Agrawal, A., Munawar, A., Nycz, C. J. & Fischer, G. S. An Approach to Modeling Closed-Loop Kinematic Chain Mechanisms, Applied to Simulations of the da Vinci Surgical System. *Acta Polytechnica Hungarica* **16** (2019).

128. Munawar, A., Wang, Y., Gondokaryono, R. & Fischer, G. *A Real-Time Dynamic Simulator and an Associated Front-End Representation Format for Simulating Complex Robots and Environments* in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2019).

129. Fontanelli, G. A., Selvaggio, M., Ferro, M., Ficuciello, F., Vendittelli, M. & Siciliano, B. *A V-REP simulator for the da Vinci research kit robotic platform* in *Int. Conf. on Biomedical Robotics and Biomechatronics (Biorob)* (2018), 1056–1061.

130. Allard, J., Cotin, S., Faure, F., Bensoussan, P.-J., Poyer, F., Duriez, C., Delingette, H. & Grisoni, L. *Sofa-an open source framework for medical simulation* in *MMVR 15-Medicine Meets Virtual Reality* **125** (2007), 13–18.

131. Talbot, H., Haouchine, N., Peterlik, I., Dequidt, J., Duriez, C., Delingette, H. & Cotin, S. *Surgery training, planning and guidance using the sofa framework* in (2015).

132. Bianchi, G., Solenthaler, B., Székely, G. & Harders, M. *Simultaneous topology and stiffness identification for mass-spring models based on FEM reference deformations* in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (2004), 293–301.

133. Morooka, K., Chen, X., Kurazume, R., Uchida, S., Hara, K., Iwashita, Y. & Hashizume, M. *Real-time nonlinear FEM with neural network for simulating soft organ model deformation* in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (2008), 742–749.

134. Roewer-Despres, F., Khan, N. & Stavness, I. *Towards finite element simulation using deep learning* in *15th International Symposium on Computer Methods in Biomechanics and Biomedical Engineering* (2018).

135. Haferssas, R., Tournier, P.-H., Nataf, F. & Cotin, S. *Simulation of soft tissue deformation in real-time using domain decomposition* Dec. 2019.

136. Meister, F., Passerini, T., Mihalef, V., Tuysuzoglu, A., Maier, A. & Mansi, T. Deep learning acceleration of Total Lagrangian Explicit Dynamics for soft tissue mechanics. *Computer Methods in Applied Mechanics and Engineering* **358,** 112628 (2020).

137. Mendizabal, A., Márquez-Neila, P. & Cotin, S. Simulation of hyperelastic materials in real-time using deep learning. *Medical image analysis* **59,** 101569 (2020).

138. Gao, H. & Ji, S. *Graph U-Nets* in *international conference on machine learning* (2019), 2083–2092.

139. Li, Y., Wu, J., Tedrake, R., Tenenbaum, J. B. & Torralba, A. *Learning Particle Dynamics for Manipulating Rigid Bodies, Deformable Objects, and Fluids* in *International Conference on Learning Representations* (2019).

140. Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J. & Battaglia, P. *Learning to simulate complex physics with graph networks* in *International Conference on Machine Learning* (2020), 8459–8468.

141. Rusu, R. B. & Cousins, S. *3d is here: Point cloud library (pcl)* in *2011 IEEE international conference on robotics and automation* (2011), 1–4.

142. Geuzaine, C. & Remacle, J.-F. Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering* **79,** 1309–1331 (2009).

143. Besl, P. J. & McKay, N. D. *Method for registration of 3-D shapes* in *Sensor fusion IV: control paradigms and data structures* **1611** (1992), 586–606.

144. Lee, J.-H., Lee, S.-S., Chang, J.-D., Thompson, M. S., Kang, D.-J., Park, S. & Park, S. A novel method for the accurate evaluation of Poisson's ratio of soft polymer materials. *The Scientific World Journal* **2013** (2013).

145. Finn, C. & Levine, S. *Deep visual foresight for planning robot motion* in *Int. Conf on Robotics and Automation (ICRA)* (2017), 2786–2793.

146. Community, B. O. *Blender - a 3D modelling and rendering package* Blender Foundation (Stichting Blender Foundation, Amsterdam, 2018).

147. Fink, L., Lee, S. C., Wu, J. Y., Liu, X., Song, T., Velikova, Y., Stamminger, M., Navab, N. & Unberath, M. *LumiPath–Towards Real-Time Physically-Based Rendering on Embedded Devices* in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (2019), 673–681.

148. Lo, S. Finite element mesh generation and adaptive meshing. *Progress in Structural Engineering and Materials* **4,** 381–399 (2002).

149. Kapoor, A., Pheiffer, T., Park, J.-h. & Kamen, A. *Deformable registration of preoperative volumes and intraoperative ultrasound images from a tracked transducer* US Patent 10,980,509. Apr. 2021.

150. Zapaishchykova, A., Dreizin, D., Li, Z., Wu, J. Y., Roohi, S. F. & Unberath, M. An Interpretable Approach to Automated Severity Scoring in Pelvic Trauma (2021).

151. Bonert, R. Digital Tachometer with Fast Dynamic Response Implemented by a Microprocessor. *IEEE Transactions on Industry Applications* **IA-19,** 1052–1056 (1983).

152. Brown, R. H., Schneider, S. C. & Mulligan, M. G. Analysis of algorithms for velocity estimation from discrete position versus time data. *IEEE Trans. on Industrial Elect.* **39,** 11–19 (1992).

153. Ekekwe, N., Etienne-Cummings, R. & Kazanzides, P. *Incremental Encoder Based Position and Velocity Measurements VLSI Chip with Serial Peripheral Interface* in *IEEE Int. Symp. on Circuits and Systems (ISCAS)* 2007 IEEE International Symposium on Circuits and Systems (May 2007), 3558–3561.

154. Sakata, K. & Fujimoto, H. *Proposal of long sampling short cycle observer for quantization error reduction* in *Int. Symp. on Industrial Electronics (ISIE)* (2010), 1919–1924.

155. Nandayapa, M., Mitsantisuk, C. & Ohishi, K. *High resolution position estimation for advanced motion control based on FPGA* in *38th Annual Conf. of IEEE Industrial Elect. Society (IECON)* (Oct. 2012), 3808–3813.

156. Zhu, H. & Sugie, T. Velocity estimation of motion systems based on low-resolution encoders. *Journal of Dynamic Systems, Measurement, and Control* **135,** 011006:1–8 (2013).

157. Kim, J. & Kim, B. K. Development of Precise Encoder Edge-Based State Estimation for Motors. *IEEE Trans. on Industrial Electronics* **63,** 3648–3655 (June 2016).

158. Merry, R. J. E., van de Molengraft, M. J. G. & Steinbuch, M. Optimal higher-order encoder time-stamping. *Mechatronics* **23,** 481–490 (Aug. 2013).

159. Hogan, N. Impedance Control: An Approach to Manipulation: Part II - Implementation. *Journal of Dynamic Systems, Measurement, and Control* **107,** 8 (1985).

160. Albu-Schäffer, A. & Hirzinger, G. *Cartesian impedance control techniques for torque controlled light-weight robots* in *Int. Conf. on Robotics and Automation (ICRA)* (2002), 657–663.

161. Tsuji, T., Hashimoto, T., Kobayashi, H., Mizuochi, M. & Ohnishi, K. A Wide-Range Velocity Measurement Method for Motion Control. *Trans. on Industrial Electronics* **56,** 510–519 (Feb. 2009).

Appendix I

Second generation da Vinci Research Kit

This appendix documents the interface to the second generation of da Vinci Research Kit (dVRK-S). While the first generation dVRK used the classic da Vinci Surgical System, the second generation uses the patient-side manipulators (PSM) from the S and Si Surgical Systems. Unlike the classic system, the PSMs in the S and Si systems have on-arm processing in the form of an FPGA, called the ESPM (Cyclone I EP1C12F256C6). Instead of passing out sensor readings (e.g. from potentiometers and encoders) directly through a 156-pin connector like in the current system, the FPGA on the dVRK-S serializes the readings. It then passes on the information through low-voltage differential signaling (LVDS). As the ESPM potentially contains proprietary interfaces to the clinical system, we made the decision to keep the ESPM source code restricted and only distribute the compiled code. I modified the PSM firmware to use an open-source communication protocol to read the signals from the arm. At the same time, I replaced the proprietary modules with open-source versions from the existing dVRK system, or wrote new modules, where possible. When deployed, the source code for these modules could be shared so researchers can see how signals are generated. The ESPM sends the signals through LVDS to a modified dVRK controller board. The dVRK controller board is modified to support LVDS communication. Instead of reading the motor feedback directly as it does in the first generation, the dVRK board would decode the same information from the ESPM before sending it to the computer. Fig. I-1 shows how these components are linked.

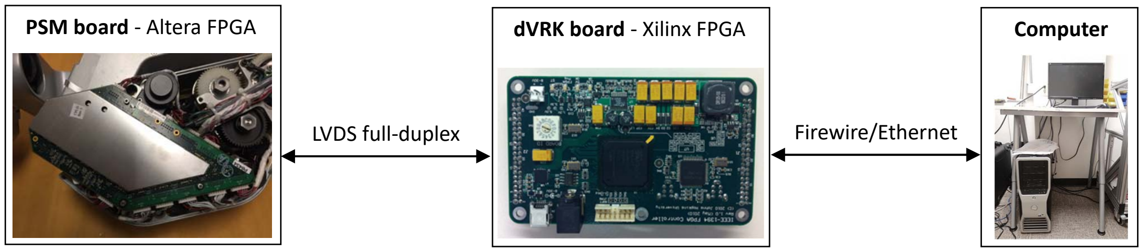


Figure I-1. Communication trace with the dVRK-S arm

The communication protocol is set to 20 Mbps LVDS in full duplex link. Fig. I-2 shows a simulation of the transmit and receive signals from the ESPM to the dVRK. Since this is in simulation, the receiver is only one clock cycle behind the transmitter. This could be longer on the robot. Each packet contains 33 quadlets, and a 16-bits CRC. The first quadlet contain an identifier to indicate the beginning of a packet, indicated by the 1 label in Fig. I-2. The identifier was arbitrarily chosen to be 32'hAC450F28. The next 32 quadlets, indicated by the 2 label, contain information about the ESPM. The quadlets are detailed in Table I-1:

Table I-1. Quadlets of the communication packet sent by the ESPM

Quadlet	Content
0	Buttons status
1-7	Joint potentiometer readings
8	ESII status (currently not used since we do not talk to the ESII)
9-15	Joint position from encoder
16	ESPM status (currently not used since we do not monitor ESPM health)
17-23	First quadlet of joint velocity information
24	Instrument ID
25-31	Second quadlet of velocity information

The potentiometer, position, and two velocity quadlets are described in Tables I-2, I-3, I-4, and I-5 respectively. The position reading is based on the Rev 7 revision.

The velocity reading is based on Rev 6 since Rev 7 requires one more quadlet to account for the running counter.

Table I-2. Potentiometer quadlet

Bit	Content
31:18	Unused
17:6	Position
5:1	Flags
0	Parity

Table I-3. Position quadlet

Bit	Content
31:25	Unused
24	Overflow bit
23:0	Counter

Table I-4. First velocity quadlet

Bit	Content
31	Overflow bit
30	Direction bit
29-22	8 least-significant bits of the last latched value
21:0	sum of the last 4 latched values (full cycle counter)

The buttons status is shown in Table I-6. There are three clutches on the arm. Each has three sensors - one for ground fault to detect whether the cable is connected, and one each of normally open and normally closed switch for redundancy. To detect the cannula, the cannula power stable should be positive before we can read the cannula ID.

After the information quadlets, there is 16 bits containing a CRC of the packet, indicated by the 3 label.

The same protocol is used to receive from the dVRK board. In motor feedback mode, the communication from the dVRK board is not used as the motor drivers are directly controlled by the dVRK board. The ESPM board scans the communication

Table I-5. Second velocity quadlet

Bit	Content
31:20	12 most-significant bits of the last latched value
19:0	All 20 bits of the value 4 cycles ago (for acceleration)

Table I-6. Buttons status

Bit	Content
31:28	Unused
27:26	Flex switch contact transition warning and fault
25:24	Cannula power stable
23	Voltage monitor error
22:21	ESII linked and fault
20:18	Cannula ID or presence switch
17	Is ECM or Sterile adapter #2 present
16:15	Instrument magnet #2 and #1 present
14:12	Instrument clutch ground fault, normally open, and normally closed
11	Instrument/camera loop back present
10:8	Flex motor switch ground fault, pressed, direction (or N.O, N.C.)
7	Is ECM or Sterile Adapter #1 present
6:4	Link 3 clutch ground fault, normally open, and normally closed
3	Debugging pin?
2:0	Port clutch ground fault, normally open, and normally closed

solely to see if there is a state change, namely to writing to the flash memory. This mode is used to write the image of the FPGA code to flash so it could be loaded at start up. To switch to this mode, the computer initiates by sending the flash command (8'hF1). The dVRK board sends it through LVDS to the ESPM, which initiates writing to the flash memory. In flash mode, the computer first sends a command (16'h0020) to erase the flash chip. Then, the computer sends two bytes of the image to be written at a time, along with the address in which to write it. Currently, the flash write is entirely managed by the computer. The ESPM board continuously reads the address sent by the computer and sends back to the computer the content. The computer checks if that matches the two bytes sent. Once the read data and the write data matches, the computer sends the next two bytes of the image. This process is somewhat time-consuming. Future work includes buffering parts of the image on both

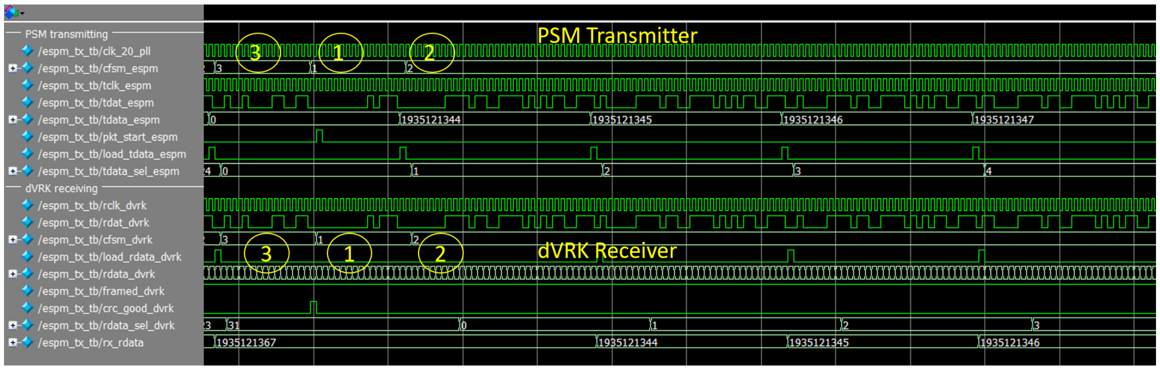


Figure I-2. Communication trace with the dVRK-S arm

the dVRK and the ESPM board and performing the write check locally to speed up the process.

Appendix II

Improved velocity estimation

This section presents a hardware-based velocity estimation method that enables improved control performance, especially for robots with low-resolution encoders and low gear reductions. The motivation for this work was provided by the da Vinci Research Kit (dVRK) [28], Fig. II-1, which combines open source electronics and software with mechanical components of first-generation da Vinci surgical robots (Intuitive Surgical, Sunnyvale, CA). The da Vinci consists of a master console with two 7 degree-of-freedom Master Tool Manipulators (MTMs) and a patient side cart with several Patient Side Manipulators and an Endoscopic Camera Manipulator. This work focuses on the MTMs, which have wrist actuators with encoders with as few as 16 lines per revolution and gear ratios as low as 16.58.

The dVRK electronics relies on field-programmable gate arrays (FPGAs) to process the robot feedback, including quadrature decoding of the encoder signals, which are transfered to a control PC via IEEE-1394a (FireWire). The PC performs joint and Cartesian-level control at loop rates in excess of 1 kHz. The FPGA firmware (Verilog) and PC software (primarily C++) are open source. During system development, it was discovered that the standard proportional-integral-derivative (PID) joint controller had stability issues for the MTM wrist actuators. Specifically, the actuators had a tendency to shake (i.e., exhibit limit cycles). The dVRK actually uses PD control (no integrator) so the instability was traced to poor velocity estimation, which affected the

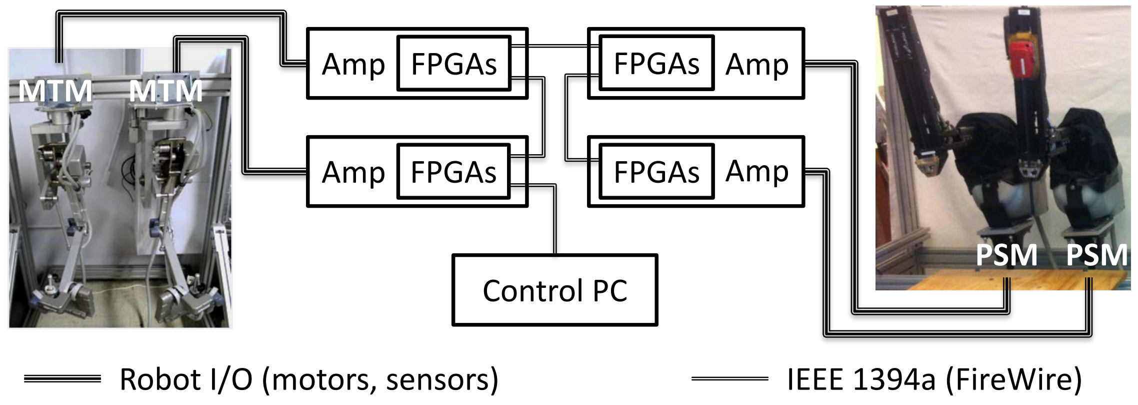


Figure II-1. da Vinci Research Kit: open source FPGA-based controllers connected to control PC via IEEE 1394a (FireWire), with direct access to motors and sensors in Master Tool Manipulators (MTMs) and Patient Side Manipulators (PSMs).

derivative term. The problem was addressed by incorporating a heuristic nonlinear gain. Furthermore, the Cartesian impedance controller exhibited stability problems for the wrist actuators, especially the final roll axis, so impedance control was disabled for this actuator.

Section A..1 provides an overview of methods for estimating velocity from quadrature incremental encoders. Fundamentally, these methods all rely on measuring the time between encoder position changes. Thus, measurements become more delayed as the joint velocity decreases. Robots with low-resolution encoders and low gear ratios experience larger delays because there are fewer counts per joint revolution. As discussed in Section A..1, this leads to a tradeoff between the responsiveness of the velocity estimation and its robustness to imperfections that introduce noise. Unfortunately, noisy or delayed measurements negatively affect control performance, including for the Cartesian impedance controller presented in Section A..2.

In Section B., we propose a novel hardware-based method that improves the responsiveness of the velocity estimation by using all encoder edges and by also estimating the acceleration. This enables the PC software to compensate for the measurement delay, leading to a solution that provides timely and robust velocity estimates. Section C. first evaluates the method on a test platform with two mechanically-coupled encoders (one high resolution and one low resolution) and then with the Cartesian impedance controller on the dVRK. The significance of this work is that it improves the control performance of the dVRK, a common research platform currently installed at 30 institutions worldwide.

A. Background and Related Work

A..1 Velocity Estimation with Quadrature Encoders

Many systems (including the da Vinci) estimate velocity from quadrature incremental encoders, which contain two channels (A and B) that produce square waves that are 90 degrees out of phase (Fig. II-2). We consider each of the following as separate events: 1) rising edge of the A channel, $A_\uparrow$, 2) falling edge of the A channel, $A_\downarrow$, 3) rising edge of the B channel, $B_\uparrow$, and 4) falling edge of the B channel, $B_\downarrow$. The joint position is obtained by counting each of these edges, with the direction determined by identifying which channel leads the other.

Velocity is estimated by calculating dx/dt, where dx is the encoder position difference and dt is the sampling interval (i.e., time between the two encoder position measurements). Typically, dx or dt is fixed to obtain either a fixed-time or a fixed-position algorithm, though some variations, such as the constant elapsed time (CET) method [151], measure time over multiple counts to achieve a minimum elapsed time.

In cases where the velocity estimation module does not have direct access to the encoder edges, it is only feasible to use a fixed-time approach, where the time between

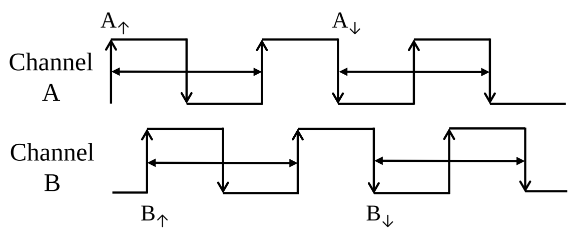


Figure II-2. Quadrature incremental encoder feedback, showing four events ($A_\uparrow$, $A_\downarrow$, $B_\uparrow$, and $B_\downarrow$) and time between consecutive occurrences of the same event.

the two position samples is based on the CPU clock. This reduces the accuracy of the time measurement because it is not synchronized with respect to the position updates. If, however, the module has access to the encoder edges, either method may be used. For the fixed-position methods, measuring the time between two edges of the same type (full-cycle measurement, as shown in Fig. II-2) increases the measurement delay but is robust to imperfections in the encoder phase (i.e., channels not exactly 90 degrees apart) and the duty cycle (i.e., channel high times not exactly equal to low times at constant velocity). In contrast, measuring the time between the two most recent edges (a quarter-cycle measurement) leads to noisy velocity measurements, where much of the noise is due to these encoder imperfections. At high speeds, the fixed-position methods suffer from time quantization errors because fewer clock ticks occur between the encoder edges (small dt), whereas at low speeds the fixed-time methods are subject to position quantization errors (small dx) and are also more affected by encoder imperfections.

Brown *et al.* compare the fixed-position and fixed-time methods for different velocity profiles as well as add higher-order terms on each scheme [152]. Most fixed-position algorithms tested were sensitive to encoder imperfections, while fixed-time algorithms were sensitive to quantization errors at low speeds. Adding higher order terms through Taylor series expansion and backward-difference expansion exaggerated these errors. While least-square-fit smooths over imperfections and quantization errors, it has bad transient response because it acts as a filter.

Low-resolution encoders and low gear ratios introduce challenges to the above methods, especially at low velocities due to the longer delays between encoder edges. We focus on fixed-position algorithms as they perform better when there is long delay between signals [153] and, in contrast to the results reported by Brown *et al.* [152], they can be implemented to be insensitive to common encoder imperfections. In practice, the definition of "low" velocity depends on the gear ratio that relates motor/encoder revolutions to robot joint revolutions. Sakata and Fujimoto use a plant model of the motor dynamics to overcome the inherent delay where the velocity measurement is always half a cycle behind due to the nature of averaging over a period [154]. To overcome delays in measurement without knowledge of motor specifications, Nandayapa *et al.* propose to add fractional steps to the position measurements in between encoder events and show that using this position for velocity estimation gives more stable results [155].

Similarly, model and non-model based methods have been examined to provide robustness against noise. Model-based algorithms can provide more accurate estimates, but require a dynamic model and a noise model, which are often inaccurate or unavailable. Zhu and Sugie [156] show that by using polynomial fitting on position data with knowledge of motor dynamics, they can accurately track the velocity for encoders with as low as 8 pulses per revolution. Kim and Kim [157] reduce this computational load by fitting on the sparser transition edges of the encoder. Non-model based algorithms are often filter-based and introduce delays. Merry *et al.* [158] examine some trade-offs in filter length and propose additional parameters such as skipping edges so filters can examine a sufficient length of data without being too memory-intensive.

The proposed method differs from the above methods first by using the rising and falling edges of both encoder channels for velocity estimation. This improves the responsiveness of the velocity estimation by a factor of four over the conventional method of measuring time between one type of edge on one encoder channel, while preserving that method's robustness to encoder imperfections. We consider only the fixed-position method because for the dVRK, the encoder resolutions and the maximum speeds with a human operator are low enough that time quantization error is not a significant issue. Because this velocity estimation still has larger delays than the quarter-cycle difference, the method additionally estimates acceleration. This enables the PC software to compensate for the measurement delay, leading to a solution that provides both the responsiveness of the quarter-cycle measurement and the robustness of the full-cycle measurement. Note that since the goal of the proposed method is to provide the best information possible from hardware, any of the above software algorithms could be adopted to further improve results.

A..2 Cartesian Impedance Control

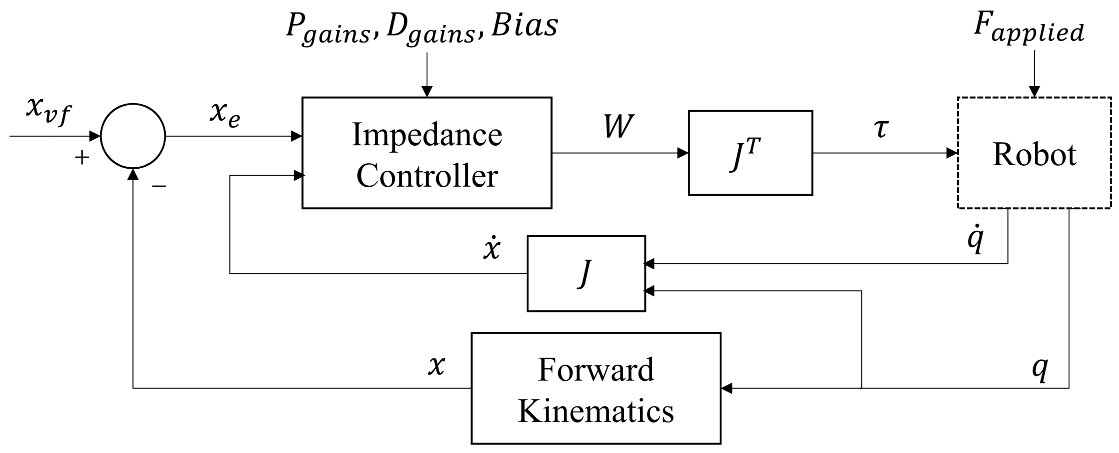


Figure II-3. Block diagram of dVRK Cartesian impedance controller

Fig. II-3 shows the implementation of the dVRK Cartesian impedance controller. The torque applied to the robot is calculated similarly to the method described in [159, 160]. This avoids calculating the inverse Jacobian by using the Jacobian transpose to calculate the torque applied to each joint.

In the figure, $(q, \dot{q})$ are the robot state (joint position and velocity) as measured by the encoders. This is converted into Cartesian position x and velocity $\dot{x}$ by forward kinematics. A force is applied to the robot based on the error in position, where the tool tip is compared to the virtual fixture position, x_{vf}, and the velocity feedback provides damping. A constant bias force is also applied. The impedance controller then calculates the Cartesian wrench, W, for translation along, and rotation around, each of the three axes as follows:

$$W = P\left(x - x_{vf}\right) - D\dot{x} + Bias \tag{II.1}$$

P and D are the proportional and derivative gains and $Bias$ accounts for constant offsets such as gravity compensation.

B. Method

B..1 Full-cycle Velocity Estimation

We observe that the velocity calculated over a quarter cycle is noisy, in large part due to the effect of encoder imperfections, and therefore calculate it over a full encoder cycle. We measure the time from one instance of an event to the next instance of the same event, as illustrated in Fig. II-2. Setting the cycle time as S and using i to indicate the event, the velocity over a full cycle (4 encoder events) can be calculated by:

$$v_i = \frac{4}{S_i} \tag{II.2}$$

B..2 Acceleration Estimation

While Eq. II.2 gives a smooth velocity estimation, the measurements are delayed because the data is estimated over a full quadrature cycle. Thus, a common approach is to assume that the currently measured velocity corresponds to the velocity in the middle of the cycle; i.e., it has a delay of a half cycle ($S_i/2$). We encountered significant delays at low velocities when calculating velocity over a full quadrature cycle compared to that of the quarter-cycle, which led to unstable controls. To estimate the velocity change over the last half cycle, we add an acceleration term, a:

$$v_i = \frac{4}{S_i} + a\frac{S_i}{2} \tag{II.3}$$

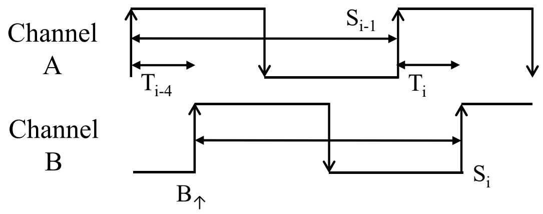


Figure II-4. Illustration of acceleration estimation. S indicates full cycles, which are robust to encoder imperfections, while T indicates quarter cycles, which are not. Acceleration can be estimated by the change between quarters of the same type, T_{i-4} and T_i, to be robust to encoder imperfections. In this example, both quarters are measured from $A_\uparrow$ to $B_\uparrow$.

The predicted velocity at i is the velocity over S_i plus acceleration over $S_i/2$, assuming constant acceleration. We estimate acceleration as a backward difference between the last two full-cycle velocity measurements. Effectively, this leads to subtraction of two quarter-cycle events that are separated by a full cycle. For example, if the last two events were $A_\uparrow$ followed by $B_\uparrow$ (as shown in Fig. II-4), the acceleration would be calculated from the difference between the last quarter and quarter that happened 4 events ago, both of which would be between $A_\uparrow$ and $B_\uparrow$, assuming no direction change. As with the velocity estimation, comparing the time between two events of the same type avoids the effects of encoder imperfections such as uneven duty cycles and phase shifts, which are a primary cause of measurement noise.

Mathematically, the change in velocity over the last two cycles, S_i and S_{i-1}, is given by:

$$\begin{aligned} \Delta v =& \frac{4}{S_i} - \frac{4}{S_{i-1}} \\ \Delta v =& \frac{4(T_{i-4} - T_i)}{S_i S_{i-1}} \end{aligned} \tag{II.4}$$

The time difference between these two velocities (assumed to correspond to the midpoints of the cycles) is given by the following, where t_i indicates the time at event i:

$$\begin{aligned} \Delta t &= \left(t_i - \frac{S_i}{2}\right) - \left(t_i - T_i - \frac{S_{i-1}}{2}\right) \\ \Delta t &= \frac{T_{i-4} + T_i}{2} \end{aligned} \tag{II.5}$$

Finally, the instantaneous acceleration is simply the quotient between the change in velocity, Δv, and time, Δt:

$$a = \frac{\Delta v}{\Delta t} = \frac{8(T_{i-4} - T_i)}{S_i S_{i-1}(T_{i-4} + T_i)} \tag{II.6}$$

Substituting Eq. II.6 into Eq. II.3 yields:

$$\begin{aligned} v_i &= \frac{4}{S_i} + \frac{8(T_{i-4} - T_i)}{S_i S_{i-1}(T_{i-4} + T_i)} \frac{S_i}{2} \\ v_i &= \frac{4}{S_i} + \frac{4(T_{i-4} - T_i)}{S_{i-1}(T_{i-4} + T_i)} \end{aligned} \tag{II.7}$$

Lastly, for the most up-to-date estimate, we can add a running counter, T_r, that measures the time since the last event:

$$v_i = \frac{4}{S_i} + \frac{8(T_{i-4} - T_i)}{S_i S_{i-1}(T_{i-4} + T_i)} \left(\frac{S_i}{2} + T_r\right) \tag{II.8}$$

Although this method avoids the noise due to encoder imperfections of duty cycle and phase, it is sensitive to quantization error because the difference between quarters is often only a few counts. This can be avoided by setting a threshold that prevents acceleration from being used at high velocities. Quantization error increases linearly with velocity as the number of counts decreases. But, delays in velocity are more significant at slow speeds, where large quantization is not a significant concern.

B..3 Implementation

To keep track of the quarter measurements, we use a queue of six 26-bit registers (Fig. II-5). The first register is a running counter that is incremented at every clock edge. The clock speed is 49.152 MHz so the counter overflows in 1.37 s. Each edge of any type adds an element to the queue, pops off the last element, and clears the first counter. Thus, the first, second, and sixth register values are used to calculate the acceleration as they represent T_r, T_i, and T_{i-4}, respectively. S_i can be calculated from summing the quarter counters T_i to T_{i-3}.

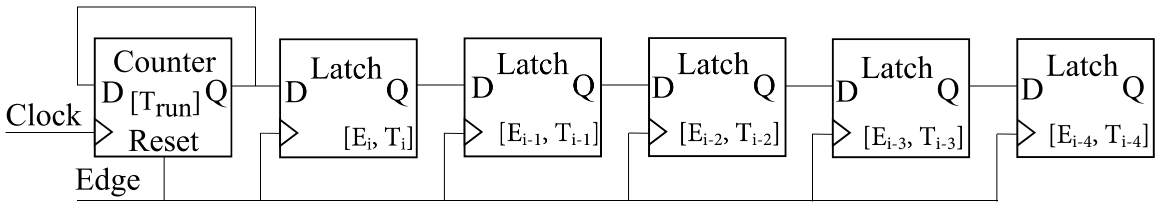


Figure II-5. Queue of running counter and quarter-cycle time measurements, $T_i \dots T_{i-4}$, for edges $E_i \dots E_{i-4}$.

The host computer issues asynchronous read requests on the FireWire bus to obtain feedback data from the FPGAs at a specified rate, generally 2-3 kHz. In the current implementation, S_i is sent as a 22-bit value, where the last 4 bits are truncated to give an effective count rate of 3.072 MHz (49.152/16). T_{i-4} and T_i are passed back as 20-bit values. Using 26-bit registers on the FPGA provides robustness against encoder imperfections because an uneven duty cycle can cause one quarter cycle to be more than 20-bits long, but the sum of four quarter cycles should still fit within 26-bits. On the PC, S_{i-1} is calculated from $S_{i-1} = S_i + (T_{i-4} - T_i)$.

We set a threshold to stop using acceleration if T_i is smaller than 2000 clock counts as it oscillates too much beyond that point. The exact threshold in velocity varies due to encoder imperfections and uneven rotation speeds.

Currently, the running counter, T_r, is not separately provided to the PC due to implementation limitations and thus Eq. II.7 is used. As a consolation, if T_r is greater than T_{i-4} the FPGA uses T_r instead of T_{i-4} when calculating S_i. This provides smoother decays in cases where the motor is decelerating, as would be obtained by using T_r in Eq. II.8.

C. Experimental Setup

We show the proposed method implemented on two hardware setups. In the first, two encoders, one high-resolution and one low-resolution, are coupled and driven by a DC motor. The gearing of the motor was removed so that the two encoders rotate at the same rate. In the second, we show the proposed method implemented on the dVRK and test its Cartesian impedance control with different velocity estimation methods.

C..1 Two Encoder Setup

Two encoders are connected as shown in Fig. II-6. The left encoder has 600 lines per revolution and we use this as the ground truth position and velocity data. The right one has 16 lines per revolution and is attached to a motor.

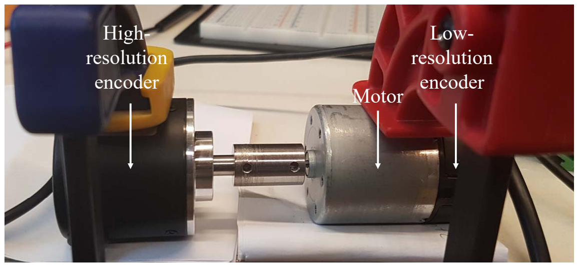


Figure II-6. Coupled encoders

To test the proposed method, we drive the motor and estimate the velocity by the proposed method, with and without acceleration, and compare it to velocity estimated by the quarter-cycle method (time between last two position changes). In our experiments, we never exceed one count difference per sampling time on the low-resolution encoder, so fixed-time algorithms are not considered.

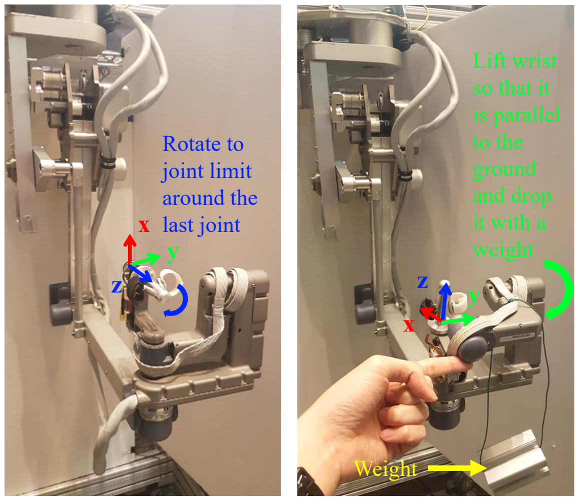
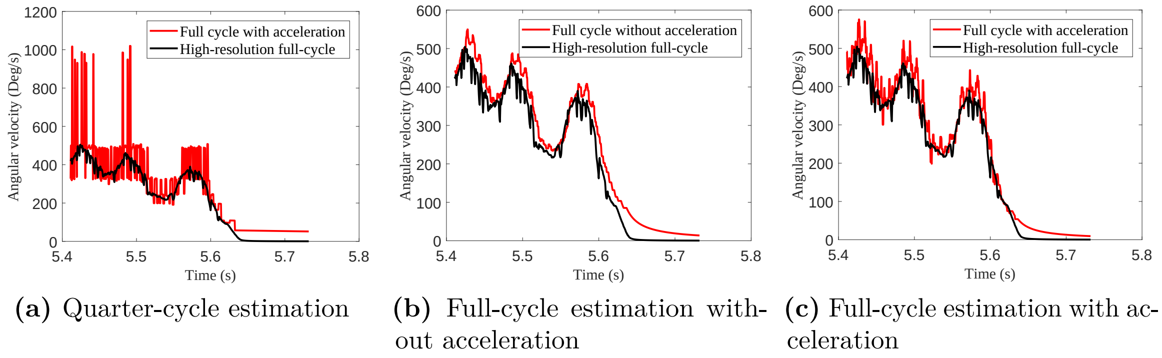

Rotate to joint limit around the last joint
x
y
z
Lift wrist so that it is parallel to the ground and drop it with a weight
z
y
x
Weight

Figure II-7. Master controller of the dVRK robot. The left image shows the joint that we rotate around the z-axis, while the right image shows the wrist being lifted around the y-axis, with a weight attached.

C..2 Cartesian Impedance Control



(a) Quarter-cycle estimation (b) Full-cycle estimation without acceleration (c) Full-cycle estimation with acceleration

Figure II-8. Comparing velocity estimation for 16 lines low-resolution encoder (red) with 600 lines high-resolution encoder (black) for different velocity estimation methods: (a) quarter cycle oscillates at high velocities, (b) full cycle is noticeably delayed even at high velocities, and (c) full cycle with acceleration (proposed method) reduces the delay but has some oscillations.

The effect of the proposed velocity estimation on control performance was tested on the left MTM of the dVRK (Fig. II-7), where the resolution of encoders ranges from 1000 lines (4000 counts) per revolution in the first four joints to 16 lines (64 counts) per revolution in the last three (wrist) joints. Furthermore, the gear ratios of the last three joints are 33.16, 33.16, and 16.58, which lead to resolutions of 0.17

deg/count, 0.17 deg/count and 0.34 deg/count, respectively.

We perform two experiments, where the first primarily exercises the last wrist joint and the second excites the larger joints and the first wrist joint. In each experiment, the robot is moved to a consistent home position and then a horizontal plane virtual fixture is created to prevent the arm from dropping due to gravity. For the first experiment, the desired (virtual fixture) orientation is set to the home orientation. The robot is set to Cartesian impedance mode and we manually rotate the last joint clockwise around the z-axis until it reaches its limit, as shown in Fig. II-7-left. We then release it and measure the step response as the robot recovers the desired orientation. Cartesian impedance gains were set at 200 N/m and 5 for linear stiffness and damping, and 0.15 N m/rad and 0.03 for torsional stiffness and damping.

In the second experiment, we set the desired orientation so that the first wrist link (closest to the robot base) is rotated by 90° around the y-axis, as shown in Fig. II-7-right. A weight of 121.5 g is attached to the wrist link while holding it in the desired orientation. We then release the link and measure the robot's position and orientation as the Cartesian impedance controller finds a new equilibrium. The two most distal wrist joints are not affected by the applied weight, and because they are orthogonal to the one supporting the weight, their possible motion has no effect on the results. We use the same impedance gains as before, except that the torsional damping is set to 0.04 to prevent the weight from hitting another link.

D. Results

D..1 High and Low Resolution Encoder Comparison

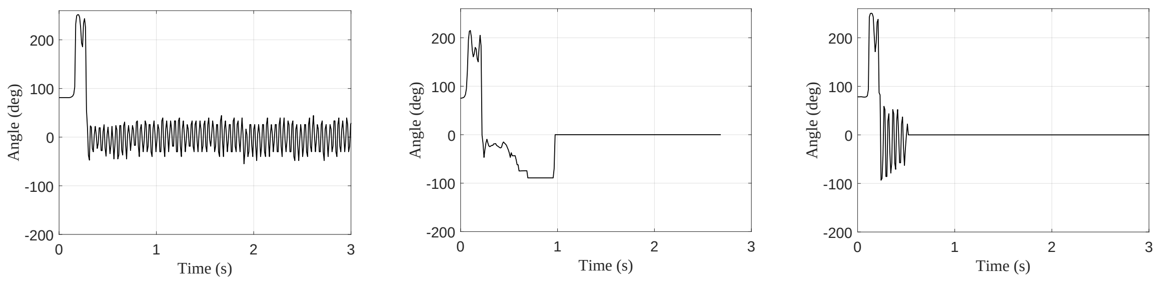
Fig. II-8 show velocity estimated by different methods on the low-resolution encoder compared to a high-resolution ground truth. The high-resolution velocity is calculated from the full cycle, without acceleration, since the measurement delay is small and thus does not require compensation based on acceleration. The RMS error between the velocity estimated by the low-resolution encoder and the ground-truth is 44.12 deg/s for the quarter-cycle method, 23.49 deg/s for the full-cycle method without acceleration, and 19.48 deg/s for the full-cycle method with acceleration. Without acceleration, we observe from Fig. II-8b that measurements, even at fairly high velocities, are noticeably delayed, and that the delay gets worse closer to 0. While the quarter-cycle velocity estimation in Fig. II-8a is the least delayed, it has large oscillations (noise) at high velocities from one-pulse differences or pulse-alterations as also observed in [161]. The proposed method in Fig. II-8c obtains a reasonable compromise between delay and noise and has the lowest error with respect to the ground truth signal.

D..2 Cartesian Impedance Control

Fig. II-9 shows the step responses when the last joint is released after having been rotated to its limit. The quarter-cycle velocity estimation never settles, while the full-cycle method has a delayed response. The proposed method (full-cycle with acceleration) is initially noisy, but settles the fastest. While the proposed method has more oscillations, in the dVRK, its shorter settling time resulted in better control performance.

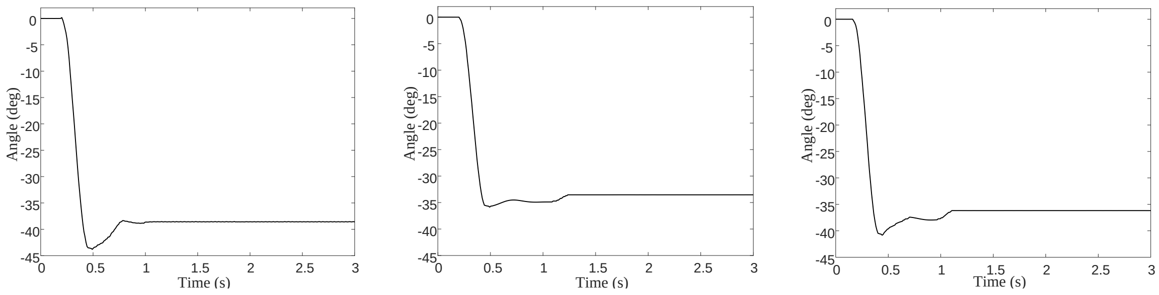
Fig. II-10 shows the step response when a weight is dropped on the third-to-last wrist joint. All methods provide stable performance because this experiment does not involve the lowest resolution joint, but the full-cycle estimation without acceleration is the slowest to respond, as expected.

Table II-1 summarizes the means and standard deviations of the settling times over ten trials for displacing the wrist around the Z and Y-axes, respectively. While not demonstrated in these experiments, the proposed velocity estimation method also improves joint control performance with a conventional PD controller (i.e., without the heuristic nonlinear gain).



(a) Quarter-cycle estimation **(b)** Full-cycle estimation without acceleration **(c)** Full-cycle estimation with acceleration

Figure II-9. Response of Cartesian impedance controller when torque applied and released about Z-axis, which primarily displaces a low-resolution, low-gear ratio joint.



(a) Quarter-cycle estimation **(b)** Full-cycle estimation without acceleration **(c)** Full-cycle estimation with acceleration

Figure II-10. Step response of Cartesian impedance controller when adding a weight (moment) around Y-axis.

Table II-1. Mean and standard deviation of settling times of rotation around z and y axes over 10 trials.

Settling time (s)	Quarter cycle	Full cycle	
		Without acc	With acc
Rotation in z	-	1.05 ± 0.62	**0.91 ± 0.31**
Rotation in y	1.76 ± 0.52	2.05 ± 0.49	**1.65 ± 0.41**

E. Conclusions

This work proposes a novel method to estimate velocity using all edges of a quadrature incremental encoder and using acceleration to overcome delays in measurement to provide better feedback for closed-loop control. The results show that this method produces a smoother, more accurate and timely velocity estimate on a low-resolution encoder. Furthermore, we showed that the improved velocity estimate leads to more stable control of a robot that has low-resolution encoders, with as few as 16 lines per revolution. The implementation (FPGA firmware and C++ software) is available open source and improves the performance of the dVRK, currently installed at 30 institutions worldwide. In addition, the proposed technique could be combined with model-based velocity estimation to further improve its accuracy, especially when a motor begins moving and there is insufficient data to calculate acceleration.