

2018-12-13

Automating endoscopic camera motion for teleoperated minimally invasive surgery using inverse reinforcement learning

Ankur S. Agrawal
Worcester Polytechnic Institute

Follow this and additional works at: <https://digitalcommons.wpi.edu/etd-theses>

Repository Citation

Agrawal, Ankur S., "Automating endoscopic camera motion for teleoperated minimally invasive surgery using inverse reinforcement learning" (2018). *Masters Theses (All Theses, All Years)*. 1252.
<https://digitalcommons.wpi.edu/etd-theses/1252>

This thesis is brought to you for free and open access by [Digital WPI](#). It has been accepted for inclusion in Masters Theses (All Theses, All Years) by an authorized administrator of Digital WPI. For more information, please contact wpi-etd@wpi.edu.

**Automating endoscopic camera motion for teleoperated
minimally invasive surgery using inverse reinforcement
learning**

by

Ankur Agrawal

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

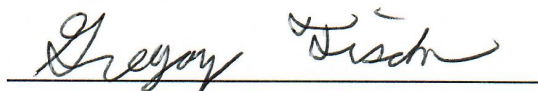
in

Robotics Engineering

by

December 2018

APPROVED:



Professor Gregory Fischer, Thesis Advisor



Professor Loris Fichera, Thesis Committee Member



Professor Jing Xiao, Thesis Committee Member

Abstract

During a laparoscopic surgery, an endoscopic camera is used to provide visual feedback of the surgery to the surgeon and is controlled by a skilled assisting surgeon or a nurse. However, in robot-assisted teleoperated systems such as the daVinci surgical system, the same control lies with the operating surgeons. This results in an added task of constantly changing view point of the endoscope which can be disruptive and also increase the cognitive load on the surgeons. The work presented in this thesis aims to provide an approach that results in an intelligent camera control for such systems using machine learning algorithms. A particular task of pick and place was selected to demonstrate this approach. To add a layer of intelligence to the endoscope, the task was classified into subtasks representing the intent of the user. Neural networks with long short term memory cells (LSTMs) were trained to classify the motion of the instruments in the subtasks and a policy was calculated for each subtask using inverse reinforcement learning (IRL). Since current surgical robots do not enable the movement of the camera and instruments simultaneously, an expert data set was unavailable that could be used to train the models. Hence, a user study was conducted in which the participants were asked to complete the task of picking and placing a ring on a peg in a 3-D immersive simulation environment created using CHAI libraries. A virtual reality headset, Oculus Rift, was used during the study to track the head movements of the users to obtain their view points while they performed the task. This was considered to be expert data and was used to train the algorithm to automate the endoscope motion. A 71.3% accuracy was obtained for the classification of the task into 4 subtasks and the inverse reinforcement learning resulted in an automated trajectory of the endoscope which was 94.7% similar to

the human trajectories collected demonstrating that the approach provided in thesis can be used to automate endoscopic motion similar to a skilled assisting surgeon.

Acknowledgements

I would like to acknowledge Professor Gregory Fischer for giving me the opportunity to work on this thesis and his invaluable support and advise throughout this work. I would also like to thank Professor Loris Fichera and Professor Jing Xiao for being on my thesis advisory committee and providing me with their insights on the project. Furthermore, I would like to thank all the lab members of Automation and Interventional Medicine (AIM) Robotics Research Lab, and in particular Adnan Munawar and Radian Gondokaryano for helping me at various stages during the project.

This research was performed using computational resources supported by the Academic & Research Computing group at Worcester Polytechnic Institute. Results in this thesis were obtained in part using a high-performance computing system acquired through NSF MRI grant DMS-1337943 to WPI, and the infrastructure for this award was supported through NSF NRI grant 1637759.

I would also like to acknowledge the Backlin Fund for funding my last credit for the thesis which allowed me to complete my thesis.

Lastly, I would like to thank my parents for supporting me and motivating me throughout my Master's and to my girlfriend for always being there for me.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 1.1 | Motivation | 2 |
| 1.2 | Overview | 4 |
| 1.2.1 | Framework | 7 |
| 1.2.2 | Goals | 7 |
| 1.2.3 | Assumptions | 8 |
| 1.3 | Contributions | 9 |
| 1.4 | Outline of the Thesis | 10 |
| 2 | Literature Review | 11 |
| 2.1 | Automation in Surgical Robotics | 11 |
| 2.2 | Intent of the user for teleoperation | 13 |
| 2.3 | Control of endoscopic camera | 14 |
| 2.4 | Inverse Reinforcement Learning | 16 |
| 3 | Algorithms for learning automation | 18 |
| 3.1 | Preliminaries | 19 |
| 3.1.1 | Neural Networks | 19 |
| 3.1.2 | Reinforcement Learning | 24 |
| 3.1.3 | Inverse Reinforcement Learning | 29 |

| | | |
|----------|--|-----------|
| 3.2 | Methodology | 33 |
| 4 | Simulation Environment | 35 |
| 4.1 | daVinci Research Kit | 36 |
| 4.1.1 | Gazebo Simulations | 37 |
| 4.2 | Endoscope Kinematics | 39 |
| 4.2.1 | Forward Kinematics | 39 |
| 4.2.2 | Inverse Kinematics | 42 |
| 4.3 | Chai3D environment | 42 |
| 4.4 | Oculus Rift Setup | 47 |
| 5 | Data Collection | 50 |
| 5.1 | User Study | 50 |
| 5.1.1 | Potential risks during the study | 54 |
| 5.2 | Data Collected | 55 |
| 5.2.1 | Video recording | 57 |
| 5.2.2 | ROS bags | 57 |
| 5.3 | Subtask Classes | 57 |
| 5.3.1 | Labeling of classes | 58 |
| 6 | Learning automated endoscope control | 61 |
| 6.1 | Inverse Reinforcement Learning Model | 62 |
| 6.1.1 | State Space | 62 |
| 6.1.2 | Action Space | 66 |
| 6.1.3 | Transition | 68 |
| 6.2 | Trajectories | 70 |
| 6.3 | Feature Set | 71 |
| 6.4 | Policy | 72 |

| | | |
|----------|---|-----------|
| 6.5 | Initial State Visitation Frequency | 73 |
| 6.6 | Validation of learned rewards | 74 |
| 7 | Results and Discussions | 76 |
| 7.1 | Subtask classification | 76 |
| 7.2 | Automation of camera motion | 80 |
| 7.2.1 | Convergence of inverse reinforcement learning | 80 |
| 7.2.2 | Reward Function | 81 |
| 7.2.3 | Value Function | 85 |
| 7.3 | Similarity of trajectories | 89 |
| 8 | Conclusion and Future Work | 93 |
| 8.1 | Conclusion | 93 |
| 8.2 | Discussion and Future Work | 95 |
| 8.2.1 | Improving accuracy of classification | 95 |
| 8.2.2 | Personalization of automated camera motion | 96 |
| 8.2.3 | Skill assessment and teaching | 96 |
| 8.2.4 | Implementation on hardware | 97 |
| 8.2.5 | Extensions of the work | 97 |

List of Figures

| | | |
|-----|---|----|
| 1.1 | The whole daVinci Surgical system developed at Intuitive Surgical Inc. | 3 |
| 1.2 | Block diagram showing the framework for collecting data and automation of endoscopic camera module for the daVinci Research Kit. | 8 |
| 3.1 | An Artificial Neural Network with two hidden layers. | 21 |
| 3.2 | A simple recurrent neural network (RNN) [49]. The circles represent the repeating module of the network and is generally a simple tanh activation function in simple recurrent neural networks. | 22 |
| 3.3 | The repeating module of the LSTM network [55] containing four layers of neurons. | 23 |
| 3.4 | Example of a grid world. The numbers on each block represent the reward for the block. The goal is to move in a manner to maximize the reward and reach the goal. | 25 |
| 3.5 | Flowchart showing how the automation of endoscopic camera manipulator (ECM) based on the master tool manipulator (MTM) motion . | 34 |
| 4.1 | Overview of the daVinci research kit including the hardware provided by Intuitive Surgical, and open source electronics and software layers as detailed in the work [67] | 37 |

| | | |
|-----|---|----|
| 4.2 | The whole daVinci Research Kit simulated in Gazebo simulator. A stereo image from the simulated ECM has been shown along with the models. | 38 |
| 4.3 | Figure showing the remote center of motion (rcm) mechanism for the Endoscopic Camera Manipulator (ECM). | 39 |
| 4.4 | Figure showing frame assignment for forward kinematics of ECM used to calculate the modified D-H parameters specified in Table 4.1. The dotted lines represent that the frames on these lines are at the same origin and are not translated horizontally. These frames can be plotted with respect to the rcm instead of the base of ECM. | 40 |
| 4.5 | Isometric View of the Simulation Environment | 43 |
| 4.6 | Side View of the Simulation Environment showing the gripper, ring, peg board and the endoscope. | 45 |
| 4.7 | Point of view of the left camera of the stereo camera on the endoscope at the initial joint configuration of the ECM showing the ring and the target peg in yellow at bottom right. If the target peg is not visible, a translucent yellow arrow pops up pointing towards the target peg. . | 46 |
| 4.8 | (left)Frames of the Oculus Rift as worn by the user with respect to the real world [72]. This frame is tracked using the sensors in the headset with respect to the world. Since the image captured by endoscope is sent to the oculus, the orientation of endoscope with respect to the oculus is shown on the right. | 48 |

| | | |
|-----|---|----|
| 4.9 | Frame transformations showing the change in orientation of the endoscope camera due to the change in orientation of the oculus. The difference between two frames depicted by “-” is actually the change in orientation mathematically given by Eq. 4.7 and multiplications denoted by “x” are post multiplication of the matrices. | 48 |
| 5.1 | A user during the user study completing the exercise using the MTM while wearing the Oculus headset. The simulation environment scene as viewed from the endoscope was streamed to the virtual reality headset. | 52 |
| 5.2 | Snapshot of the video from one of the user studies showing the arrow pointing towards the target peg whilst the user is approaching to pick up the ring. Additional information regarding the simulation time and frequency of graphic and haptic update is shown at the bottom of the screen and the subtask label for classification is shown at the top. | 53 |
| 5.3 | Figure showing the pin cushion distortion due to the lenses of Oculus Rift and countering it using barrel distortion. | 55 |
| 6.1 | Figure showing the state of the gripper shown in the image of the camera view. The state of the environment is considered to be the gripper position in the image axis and the velocity of the gripper movement represented in the polar coordinates. | 63 |
| 6.2 | Histogram showing the variation of z_{gt}^s for the trajectories obtained during the data study along with the Normal Distribution curve (in red). | 69 |

| | | |
|-----|---|----|
| 6.3 | Histogram showing the frequency with which states were visited in the collected data trajectories. Four histograms are plotted for each variable of the state along with the probability distributions (in red). | 74 |
| 7.1 | Graphs showing the accuracy and loss for the implemented neural network evolve over the number of epochs for train and validation data sets. A training accuracy of 74.5% and a validation accuracy of 71.3% was obtained. | 77 |
| 7.2 | Confusion matrix for the subtask classification generated for a completely unseen data set. It can be seen that the accuracy is reduced due to confusion between subtasks 3 and 4. | 79 |
| 7.3 | Rewards obtained by the expert trajectories evolving with the number of iterations of the IRL for all 4 subtask classes in order from left to right. | 82 |
| 7.4 | Reward function computed by the inverse reinforcement learning algorithm for the subtask 1. This figure shows the reward function for iteration 0 (left), 1 (center), 10 (right). Each row shows the reward function for the 4 classes of subtasks. | 83 |
| 7.5 | Value function computed by the inverse reinforcement learning algorithm for all subtasks (arrange by the row). Columns show the value function for iteration 0, iteration 1 and iteration 10 of the algorithm. | 86 |
| 7.6 | Final value function obtained from IRL plotted for different speeds of gripper motion in the negative x direction: 0, 0.01, 0.05, 0.1 units in order in columns for all subtasks (rows). | 87 |
| 7.7 | Final value function obtained from IRL plotted for different directions of motion of gripper at speed of 0.05 units: right, up, left, down in order in columns for all subtasks (rows). | 88 |

| | | |
|-----|--|----|
| 7.8 | Comparison between the recorded human trajectories with the automated trajectory in the joint space of the endoscope for the same gripper motion data. | 90 |
|-----|--|----|

List of Tables

| | | |
|-----|--|----|
| 4.1 | Modified Denavit-Hartenberg parameters for the endoscopic camera manipulator (ECM). | 41 |
| 4.2 | Table showing different parameters of the stereo camera in the simulation | 46 |
| 5.1 | Table describing the variables that were recording during the user study. | 56 |
| 6.1 | Table showing the number of trajectories for each of the subtasks. . . | 71 |
| 6.2 | Table showing the initial state value frequency distribution for each variable of the state space. | 73 |

Chapter 1

Introduction

Minimally invasive surgeries or laparoscopic surgeries [1] traditionally involve surgeons using small incisions to the affected area to put surgical instruments through for operation including an endoscope. This endoscope provides a video feed of the operating area on a monitor. Since only a small incision is made in laparoscopic surgeries, the recovery time for patients reduces significantly compared to the open surgeries. Moreover, since only specific affected parts of the body are exposed, other neighboring body parts do not get infected. However, some disadvantages still exist like limited range of motion of instruments, no depth perception of the operated area, tremor magnification due to instruments, and longer operating hours to name a few. Introduction of robotic laparoscopic surgeries tackled most of these disadvantages and a detailed review and history of robotic surgery is presented in [2], [3].

Advancements in the field of medical robotics has enabled these laparoscopic surgeries to be teleoperated using a master-slave system. This teleoperation has several advantages including tremor reduction, increased accuracy of the operation and minimal recovery time for the patient. Moreover, it provides better ergonomics

for the surgeon and an increased dexterity due to the presence of articulated wrists for most of the surgical instruments. Furthermore it provides a 3D stereographic view of the operating area rather than constraining the view to a 2D monitor during the surgery. However, when compared with traditional laparoscopic surgeries, some disadvantages still exist. One major disadvantage is that the operating surgeon is tasked with endoscope control in addition to the control of surgical instruments. This leads to an added task load for the surgeon and disruptions while performing the surgery to adjust the view points of the endoscopic camera.

The thesis documented here presents a way to learn the endoscopic camera motion during a teleoperated laparoscopic surgery using reinforcement learning. This chapter introduces the idea of automation in medical robotics and the pertaining literature, provides motivation for the thesis, a brief overview and the architecture used during the thesis.

1.1 Motivation

In traditional laparoscopic surgery, a co-surgeon is required to control the endoscope. Generally, expert surgeons are also expert camera assistants as they provide optimal camera views for the surgeon and sometimes help guide the procedure. This is how a teaching surgeon guides a learning resident during a certain procedure. However, in most cases a junior resident is the one controlling the camera while the expert surgeon performs the surgery. This enables the resident to learn the optimal view points for the procedure by a constant verbal feedback from the surgeon and in time become an expert surgeon himself. Since a learning resident can learn the optimal camera placements that do not cause hindrance to the surgeon and provide the best views for the particular procedure by reinforcing the feedbacks from the surgeon,

INTRODUCTION

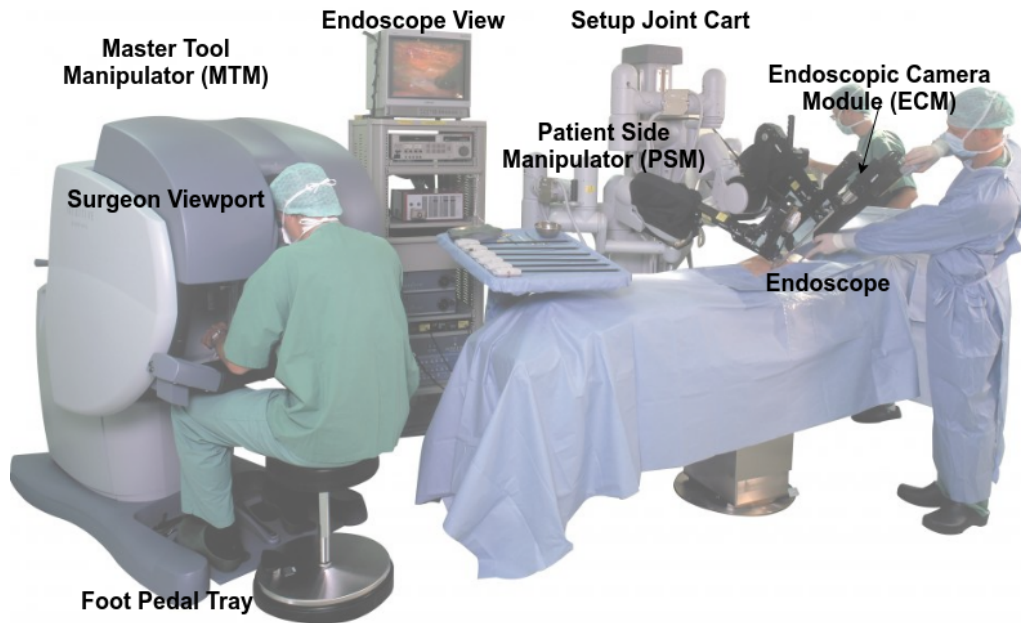


Figure 1.1: The whole daVinci Surgical system developed at Intuitive Surgical Inc.

a similar learning can be applied to autonomously control the endoscope during a teleoperated minimally invasive surgery.

In current systems for teleoperated minimally invasive surgeries, the operating surgeons themselves control the camera by pressing on a foot pedal named “Camera”. While the foot pedal is pressed, the instruments are not allowed to move hence pausing the surgery while the surgeon looks for a better view.

Fig. 1.1 shows the daVinci surgical system by Intuitive Surgical Inc. This system is the most widely used surgical robotics system for teleoperated laparoscopic surgeries. The figure shows the surgeon console and the slave manipulators that operate on the patient. The surgeon views the operation from the endoscopic camera in the scope of the surgeon console and the same view can be seen on the monitor as well. The surgeons are generally manipulating the instrument arms or patient side manipulators (PSM). These manipulators are moved using the Master Tool Manipulators (MTM), which are two (left arm and right arm) 7 degrees of freedom articulated

robotic arms (redundancy is provided for better ergonomics) incorporating a gripper, in the surgeon console. If they need to change the view of the operating area, they press on a camera pedal which locks the instruments from moving and activates the endoscopic camera manipulator (ECM) and it can be moved again with the MTMs to a desired viewing angle. Sometimes due to this added procedure to manipulate views, the surgeons may settle for a suboptimal view. This is not a favorable condition as the study in [4] shows the difference between using dual view of the operating region versus a single view. The study concludes that the orientation alignment errors required for tasks like knot tying were high for a single view and there were more collisions of the instruments detected for the same case. Hence, it shows camera viewing angles are important for the performance of a surgery and a need for a system that can provide the optimal camera views.

Moreover, having an autonomous endoscopic camera motion which is intelligent to understand the intent of a surgeon could be a stepping stone in automating surgeries in the future. The most pertaining reason as to why surgeries cannot be automated currently is that there could be a whole lot of situations for a given procedure and would require correct actions all the time for the success of the surgery. This would directly impact the outcome of the surgery and is absolutely critical to have a reliable system. With an intelligent camera, there is only a minimal risk to the operation itself but the approach to making it intelligent can be applied in the future to automate surgeries.

1.2 Overview

This thesis presents a way to automate the endoscopic camera motion for a teleoperated surgical robot, specifically daVinci surgical robot. The main objective of the

INTRODUCTION

thesis is to develop an approach for automated camera motion such that it is reactive and responsive to what the surgeon is trying to do, thereby adding an intelligence to it. For such an intelligence however, it is important to know what the surgeon is doing or at least their intention. This intention affects the way the camera should be moved and in fact is highly dependent on it. For instance, if the surgical robot is used to pick up and place rings on pegs, picking up would require the camera to be focused on the instrument tips picking up the ring. After, the camera has to be zoomed out and moved accordingly to show the peg board and while placing, the camera should again be zoomed in on the peg on which the ring is being placed.

This particular task of picking and placing a ring on a peg was studied in this thesis for automating the camera motion. Inspiration to use this task as a basis for study comes from the fact that this task amongst others are the most basic tasks used to train surgeons for the robotic surgery as suggested by [5]. In this paper, the authors suggest a curriculum including five tasks that each surgeon should go through to master laparoscopic surgery. Amongst these tasks, it was thought that this particular task of pick and place involved most amount of camera movement and was therefore chosen an example to demonstrate an approach towards automation of camera.

As can be seen from the example above, the camera motions depend heavily on the activity that is being performed and has to be reactive to the actions of the user performing those tasks. Hence classifying intentions of the user is to identify the subtask the user is performing and is imperative for a truly intelligent camera control. Moreover, this thesis develops an algorithm without the computationally expensive process of image processing and uses only the kinematic data of the surgical arms. Since the video feed is not processed, information regarding the location of objects being manipulated is not exactly known however, this information is hidden

INTRODUCTION

in the movement of the instruments and hence it is hypothesized that the sequence of movements of the instruments is enough information to classify the intent of the surgeons and provide enough information to make a reactive camera control. Though the information of how the camera should be moved is not encoded in this kinematic data.

To collect this data on how the camera should be moved for a particular task, a user study was conducted where users would perform a task of picking and placing a ring on a peg. An interactive environment was developed which was compatible to use with virtual reality headset (Oculus Rift CV1) and the users were asked to wear this headset while performing the task. The motion of the headset were mapped to the motion of the simulated camera so that the users view change while they do the task and we get the information of where and how people look while picking and placing a ring on a peg. Additionally, a foot pedal was used to control the zoom in and out of the simulated camera and this could be done even when the instruments are moving. This provided a template data on how the camera should be moved according to the movements of the users.

Machine learning algorithms were then devised to extract both the intent of the user given their movements and learn the optimal camera views according to these movements. To classify the intent based on the movements, the task was divided into subtasks and these subtasks were labeled while the users were carrying out the tasks. The subtasks were classified into four categories: Approaching the ring to pick it up, picking up the ring, approaching where to place the ring and placing the ring on the desired peg. These categories were decided because the behavior of the camera would be very different in each of these categories.

A recurrent neural network, with a long short term memory (LSTM) [6] hidden layer was now used to learn the classification based on the supervised data set thus

created. These LSTM layers are so effective since they have the ability to take in the events of the previous time steps (as all recurrent neural networks) but also to selectively forget the previous inputs when required (for instance, when a subtask is switched). Details about the models and why they were chosen in particular have been presented in further chapters.

Inverse reinforcement learning or learning from demonstration was used to learn the desired behavior based on the motion data captured using the virtual reality headset. Given a desired motion, this type of learning learns how rewarding or penalizing an action taken could be from a particular state. Preliminaries about the algorithm and its application for optimizing camera views is presented in further chapters.

Based on this overview, the goals and assumptions have been summarized and stated below.

1.2.1 Framework

An overall view of the framework used in this thesis to collect data of where users see while performing the task and how it is used to learn the behavior for automation is shown in Fig. 1.2. The framework is designed such that it can be used for both the hardware and simulation environments, albeit data can only be collected using the simulation. As mentioned earlier, the current system for teleoperated surgery does not allow the camera and tools to be moved simultaneously and hence a virtual reality device (Oculus Rift) was used to collect a baseline on which the learning algorithms can be built.

1.2.2 Goals

There are two primary goals of the thesis:

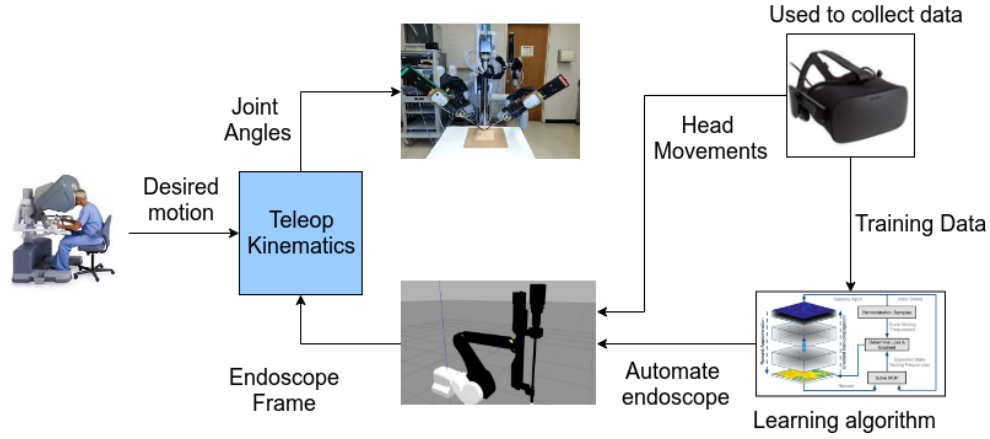


Figure 1.2: Block diagram showing the framework for collecting data and automation of endoscopic camera module for the daVinci Research Kit.

- **Detect the intent of the user** : To detect what the user is trying to do, for instance are they trying to pick up the ring or place it. In other words, it is identifying the subtask that the user is performing at any given time.
- **Learn the optimal camera motion** : Learn how the camera should be moved at each instance based on the subtask the user is performing. This only rely's on users motions and does not depend on where the other objects in the environment are.

1.2.3 Assumptions

There are several assumptions that were made to simplify the task of automating endoscopic camera movements:

- It is assumed that there is only one unique optimal camera view for a given scenario. However this might not be true since different people have different preferences on how they would like to see while performing a certain task. Different surgeons might prefer different viewing angles during the surgery. This assumption can however, be justified by the fact that the endoscope

movement is very constrained since it is only a 4 degree of freedom arm (instead of a 6), and only be inserted and rotated about a pivot during the surgery. Moreover, this thesis performs a user study and the resulted behavior is a generalized behavior of all the participants. Therefore if certain preferences are required, the algorithms presented here can be trained using a particular data set containing the preferred behavior.

- Users while using the virtual reality headset were assumed to focus their attention on the center of the frame at any given time. This might not be true if the users move their eyes independently of the head motions. But since, there is no eye tracker installed in the headset used, the gaze of the user is not tracked and recorded.
- Human trajectory (considered “expert trajectories” for IRL), i.e. is the trajectories obtained from the user study is the optimal trajectory and the users at all times are aware of the task to be performed. However, the learning algorithms used in this thesis account for the variance in human performance by using exponential and Gaussian functions.

1.3 Contributions

- Developed a 3D simulated virtual reality environment to pick and place a ring on a peg.
- Interfaced the Oculus Rift CV1 with the simulated environment to give an immersive experience.
- Performed an user study to collect an open source dataset recording desired camera movements (using Oculus Rift) with the movements of the master tool

manipulator.

- Developing machine learning models for classifying subtask of the task at hand (for eg. picking an object or placing it) to better automate the motions of the endoscope based on these subtasks.
- Using inverse reinforcement learning to learn a desired behavior to automate endoscope based on the demonstrations obtained from the user study.
- Validating the learned behaviors by comparing the obtained automated camera trajectory with a human trajectory from the user study.

1.4 Outline of the Thesis

This thesis is organized as follows: Chapter 2 reviews the related work on automation for surgical robots, camera automation in general, endoscope camera automation and inverse reinforcement learning. Chapter 3 presents the background on the algorithms that are used in this thesis and introduces the terms and variables to the readers. Chapter 4 details the current system architecture, kinematics of the ECM and the simulation environment built for the thesis and how the Oculus was interfaced with the simulation. Chapter 5 describes the user study and the data that was collected during the study. In Chapter 6, the implementation details of the algorithms described in Chapter 3 are discussed thoroughly. This includes the defining variables specific to the task of automation of endoscopic camera. Chapter 7 discusses the results of the implemented algorithms and Chapter 8 provides the future works that can be developed on this thesis.

Chapter 2

Literature Review

In this thesis, the aim to provide an approach in achieving a camera control that anticipates the user's intent by determining the subtask that is being performed and define the behavior of the camera according to the subtask. This chapter of the thesis provides a background of the work done in related fields and puts the thesis in context with the current literature. Since the thesis has a lot of facets involved like surgical robots, camera control, automation and machine learning, human-robot collaboration via teleoperation, this chapter has been divided into different sections pertaining to literature of each of these facets.

2.1 Automation in Surgical Robotics

Automation during surgeries using robots is a very challenging task since surgical procedures performed are very complex and require experience for adapting to various situations during the procedure. Additionally, same operations also differ from each other and vary for different patients. Therefore, a fully autonomous robot which can carry out a procedure from start to end is currently far fetched, though potentially can make health care consistent around the world. However, surgical

procedures can be broken down into smaller tasks or subtasks that can be automated. For instance, the Smart Tissue Autonomous Robot presented in [8], [9] is an autonomous robot performing suturing under the supervision and guidance of a surgeon. The surgeon can interact with the robot to guide/correct it whenever a wrong motion is done using high level commands. The algorithm uses a near infrared fluorescent imaging system for sensory inputs to the algorithm for task automation. On the same robot, the work done in [10] shows autonomous precise tumor cutting. This autonomous task was demonstrated to be better in efficiency than the surgeons resulting in lesser damage to the surrounding tissue. An extensive review of autonomous and semi-autonomous systems has been provided by Moustris et. al. [12].

Das et. al. [11] provides another detailed review of autonomy in robotic surgeries. The authors discuss the potential advantages (greater precision, stability) and disadvantages (lack of judgement) of autonomy. Further, the authors classify levels of autonomy for surgical robots into four broad domain: direct control, shared control, supervised autonomy and full autonomy.

Recent research at the University of California, Berkley shows automation using machine learning where different subtasks were automated using deep learning. In [13], [14], authors have successfully demonstrated cutting of deformable objects like a gauze. The authors have used deep reinforcement learning to learn where an instrument must hold the object to generate a particular amount of tension in the body to facilitate cutting.

2.2 Intent of the user for teleoperation

For any human-machine collaborative automation to work for robotic surgery, it is imperative to know the intention of the human operating or to classify the subtask the human is performing so that the machine can respond according to it. An example of such a task has been shown in [17], where the authors automate a subtask while other subtasks are being performed by the human for suturing. They use Hidden Markov Models (HMMs) modeled using Gaussian functions with expectation maximization to detect what subtask the human is in and when it is completed so that the next subtask can be automated. It has been demonstrated on the daVinci surgical robot. The approach provided in paper is particularly applicable to the objective of this thesis and was a motivation for the work done. However, a different method to classify subtasks using recurrent neural networks have been used in this thesis. The HMMs are very computationally expensive since they use dynamic programming algorithms to compute. The study done in the paper [16] shows the comparison between the two algorithms HMM and Long Short Term Memory (LSTMs) for a classification. The authors concluded that the LSTMs output a better accuracy when more data is available however the HMMs perform better to a change in stimuli and therefore should be used when early response is required. Since a user study was done during this thesis, there was enough data present and therefore LSTMs were used for classification in place of HMMs.

This intent or classification of subtasks has also been referred to as surgical gestures by [18]. In this paper, the authors analyze the surgical motions during suturing and classify the motion into 8 “surgical gestures”. To segment these surgical gestures, a temporal feature is created by concatenating neighboring time samples and a Bayes’ classifier is used to classify motions into gestures. An open source,

shared dataset named JIGSAWS for analysis of surgical motion and skill assessment has been released [19] where different surgical tasks such as suturing, knot-tying, and needle passing have been recorded. A more comprehensive analysis on gesture segmentation and recognition using the JIGSAWS dataset has been done in [20]

This view of intent recognition has also been supported by [15] where the authors suggest an interface for a collaborative/shared autonomy control for teleoperated robots in general. This interface would gather information and predict the operators intent using this information. The authors also argue that teleoperation with shared/collaborative autonomy can be a way towards robots learning the skills of experts by biasing towards the actions of the expert. Hence, learning the operators intent is a stepping stone for achieving autonomy for various complex tasks such as surgeries.

2.3 Control of endoscopic camera

Even before the advent of daVinci surgical robot, robotic arms have been used to replace the assisting surgeons for operating the endoscopic camera. Some of the earliest works focused on reducing the stress on assisting surgeons by using static mechanical structures to hold the endoscopic cameras [21]. However, these static structures were infeasible for most of the laparoscopic surgeries and therefore advanced robotic arms such as TISKA [22] were developed. This mechanical arm was kinematically designed such that it has an invariant point which does not move during the operation and can be controlled by the operating surgeon using foot pedals. This idea of a remote center of motion was crucial to the development of the surgical robots. The development of different endoscopic modules and their mechanisms, kinematics was reviewed by [23] detail.

Advancements from these mechanical holders led to the development of robots such as AESOP [24] which is manipulator with 4 actuated and 2 passive joints using a SCARA like kinematic structure. This endoscopic arm can be used via voice control and joysticks. Further research with AESOP by Ali et. al. [25] led to automating the control of the robot using eye gaze tracking from a video feedback monitor. The algorithm ensured that the camera was moved in such a way that the camera viewpoint is centered at the user's gaze. Although this provides a hands free approach to controlling the camera, the camera itself is not intelligent and is simply reacting to the sensory input of eye tracking.

Most widely implemented methods for achieving autonomy in camera control during laparoscopic surgery involve tracking of instruments in the images as can be seen in the works [26], [27], [28]. Generally, the instruments are detected in the image using color segmentation of the instruments or by using colored bands at the end of the instruments to track them. Some researches [29], [30] uses the information of the insertion point of instruments and their shape to increase the efficiency of tracking and reduce the computational power needed to track the instruments thereby making it feasible to track them in real time. Kinematic data of the manipulator arms too can be used if available to track the instruments when registered with respect to the camera. Once the instruments have been tracked, the camera is controlled to follow some specific rules.

For instance, the camera control algorithm described in [33], [34] uses the tracking information to move the camera such that its viewpoint is at the centroid of the two instruments and zoom is determined by the distance between the instruments. This work was done for the daVinci surgical system. A similar work for the AESOP system was done in the thesis [32]. More systems using visual servoing by instrument tracking have been reviewed in [35], [36]. However such automated systems

assume that the instruments are the sole focus of the surgeon which might not be true in real dynamic scenarios where the state of other objects change constantly.

In the survey on automating camera viewpoints [37], the authors provide a detailed review on the current systems and method used to achieve automation of endoscopes. They suggest that for an autonomous camera control to be intelligent and successfully replace an assisting surgeon, the camera needs to be a combination of reactive and proactive elements by knowing and detecting the intent of the surgeon. [31] describes a novel bending endoscope that can be controlled by voice commands and also in autonomous mode. Colored bands are used to track the instruments. However, the algorithm developed after consultation with the surgeons takes into account the stage of the surgery and accordingly decided whether to move or be static.

In this thesis, the aim to provide an approach in achieving a camera control that anticipates the user’s intent by determining the subtask that is being performed and define the behavior of the camera according to the subtask.

2.4 Inverse Reinforcement Learning

Inverse reinforcement learning was introduced by Ng et. al. [38] for applications where the reward function was difficult to define but an agent could still be automated by giving expert demonstrations of the task. Since then there have been several work in the field and many algorithms have been developed such as [39], [40], [41] to cite a few. [42], [43] provides a survey of the current inverse reinforcement learning algorithms and the challenges encountered by them. Probably the most popular algorithm used by the community would be the maximum entropy inverse reinforcement learning algorithm introduced by Ziebart et. al. [44].

This algorithm is used for the thesis and is explained in detail in the next chapter. Another algorithm considered for the thesis was [45] which models the inverse reinforcement framework as a two player game for cooperative task handling between two autonomous agents.

Inverse reinforcement learning is also referred to as inverse optimal control in [46] where the authors derive the mathematics to apply inverse reinforcement learning for continuous state and action spaces. In all of the papers cited here, the algorithms developed are tested within two specific environments: gridworld and autonomous driving. The algorithms developed are tested for different behaviors in each environment (for eg. aggressive and evasive driving).

In the autonomous driving environment, an autonomous agent is controlled to drive in traffic and follow lanes. The traffic is independent of the actions of the autonomous agent. This environment is very similar to the task of automating endoscope. For this task, we have instruments that are independently moved with respect to camera and the camera is to be automated to follow a specific behavior. This similarity in the situations was used as inspiration for using inverse reinforcement learning to automate the endoscopic motion.

Chapter 3

Algorithms for learning automation

In this chapter, basics of recurrent neural networks, reinforcement and inverse reinforcement learning will be introduced since these algorithms are used during this thesis. This chapter aims on providing the necessary and sufficient information to the readers to be able to understand the approach taken in the thesis to automate the endoscope camera. Furthermore, this chapter will discuss how these algorithms will be used to tackle the problem at hand and a mathematical problem formulation will be done.

Reinforcement learning or learning by experience is a branch of machine learning that penalizes bad actions and rewards good actions. In this type of learning, desired sequence of actions are learned by a given reward function that quantifies how good or bad an action is. This type of machine learning differs from the supervised learning where each output has a “target” output and the prediction is evaluated against that target. The information regarding the target is not available in this class of machine learning algorithms but only a minimal information regarding whether an

action is good and bad and the degree to which they are good is known. Examples of the use of reinforcement learning has been to have an autonomous agent play games similar to pacman where states and rewards are easy to define. These are used by some researchers for surgical task automation as well as discussed in the previous chapter to automate gaze cutting. This thesis uses this method to automate endoscope by extracting rewards from a user data using inverse reinforcement learning (IRL).

IRL also known as inverse optimal control or learning by demonstration, on the other hand learns the reward function given a expert motion data. This type of learning is generally used to mimic human behaviors, and most of the literature involves using this for autonomous driving on a highway. The different behaviors in this example would be different types of driving like aggressive, passive, tailgating, etc. In this thesis, this algorithm would be used to control endoscope camera and mimic the behavior of a human by knowing their point of views for different situations.

The approach presented in this work classifies the input data which is a time series data into subtasks using neural networks to know the intent of the users and then an inverse learning model is trained for each of the subtasks that outputs a policy deciding the action to be taken based on the state of the environment. These terms are better defined in the section below.

3.1 Preliminaries

3.1.1 Neural Networks

Artificial neural networks or simply neural networks [47] are biological inspired mathematical structures that are used to estimate a function between input and

output variables. These neural networks have a functional unit called a “neuron” that is repeated in a layer, with multiple layers and are interconnected using weights to form a network. Fig. 3.1 shows a typical neural network with two hidden layers. In the figure, the circles seen are the neurons where each neuron from one layer is connected to each layer of the next layer. The arrows represents the weights that the output of each neurons get multiplied with and act as inputs for the succeeding layer of neurons. By knowing all the optimum weights, a relation between input and output can thus be calculated.

The neural networks use a supervised learning algorithm like a back propagation algorithm. It is called as supervised because while using the neural networks there is a target data sample available for every input data sample. Once a prediction is made by the network, the error (typically root mean squared error) is calculated as the difference between the predicted output and the target output. The objective is thus to define “optimal” weights that minimize this error for the data set used for training. The gradient of the error function is calculated and the error is back propagated through the layers to change the weights in the gradient decent direction and the weights are updated. This updates the predicted value to reduce the error with the target values. Thus the target data sample “supervise” the prediction made by the neural network and guide the prediction. In contrast, as discussed in the next section, reinforcement learning does not have a target data but only a feedback quantifying how good an action performed by agent is. There is no information providing what the action should be. The algorithm finds the best action over time for a given state of the system to move into a new state.

Even though neural networks are widely used for pattern recognition, classification, regression, they are not efficient in decoding pattern for a time based series of events. This is due to the fact that neural networks have no “memory” to remem-

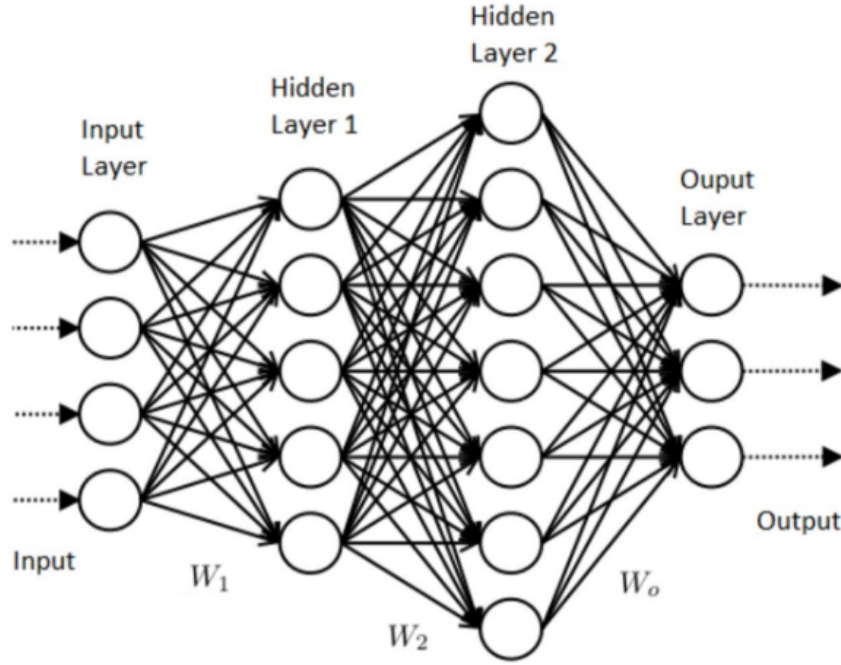


Figure 3.1: An Artificial Neural Network with two hidden layers.

ber the previous output and use it to impact the next output. Even though the inputs from previous times can be stacked to make a multi-dimensional input, the information about the last prediction cannot be known for the current prediction. This is essentially required for a time series data where decisions are based on earlier decisions.

With respect to the task of picking and placing for the work presented here, the subtask such as picking up the ring cannot take place if the previous task predicted was placing the ring on the target peg. To place the ring, the ring should already have been picked. However, the neural network does not have the information of the previous predicted value and thus could predict “picking up ring” after “placing the ring” was already predicted. This is very undesirable and would result in a low accuracy model.

For this, a recurrent neural networks (RNNs) [48] can be very effective. In

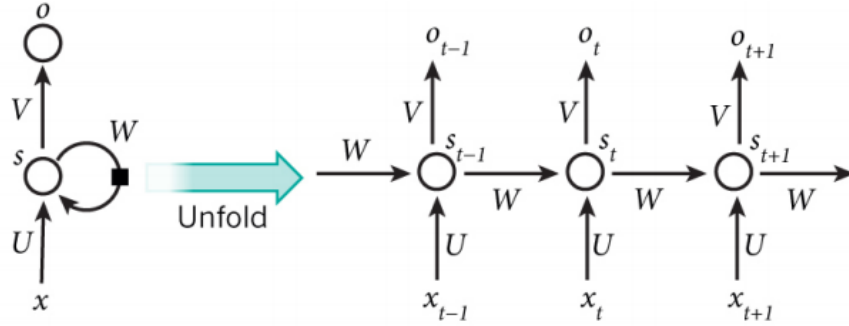


Figure 3.2: A simple recurrent neural network (RNN) [49]. The circles represent the repeating module of the network and is generally a simple tanh activation function in simple recurrent neural networks.

RNNs, the neurons are interconnected with each other to form loops and therefore can preserve an internal state which can be used to process the next output. Due to this feature, RNNs can be used for temporal dynamic behaviors and can be used in tasks like continuous handwriting recognition [50] and speech recognition [51], [52]. [53] proved that RNNs can successfully be used to generate any trajectories for a n -dimensional dynamical system. Fig. 3.2 shows a simple RNN. In the figure, the circles represent a neural network and not a single neuron.

Long Short Term Memory networks (LSTMs) first introduced by Hochreiter and Schmidhuber (1997) et. al. [54] are RNNs with some additional features and are capable of learning long term temporal dependencies. The principal behind the LSTMs are that they have different gates that modify the internal cell state. These gates are simply a combination of neurons of the neural network. Fig. 3.3 shows the structure of the LSTM networks. From this figure it can be seen that LSTM has four layers or gates of neurons performing specific functions.

The workings of the LSTM primarily depend upon the cell state that is the horizontal line at the top of the cell. This cell state carries the temporal information and is changed and used by the different gates of the LSTM. The information carried in the cell state at time t is denoted by C_t . The first layer in the LSTM network

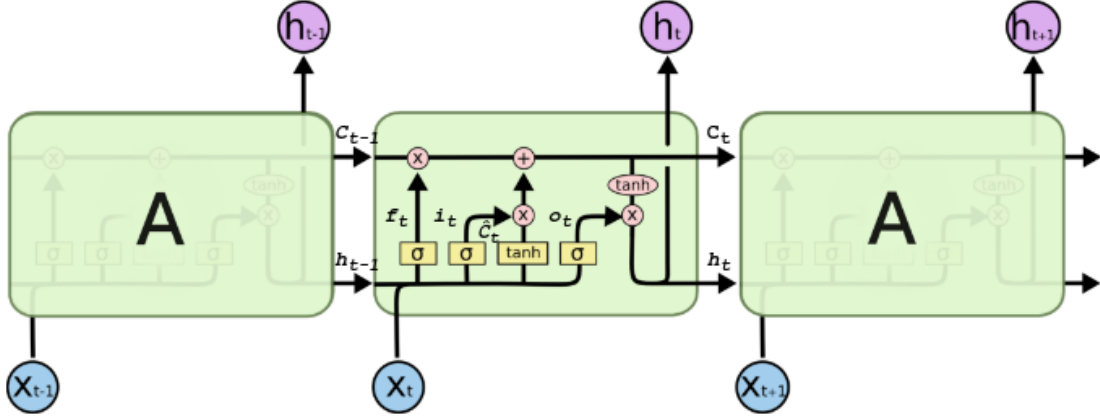


Figure 3.3: The repeating module of the LSTM network [55] containing four layers of neurons.

is known as the “forget layer” where based on the input and last predicted output, the sigmoid function calculates a number f_t between 0 and 1 and is multiplied by the cell state. This decides how much of the information of the cell state is kept with zero representing none and 1 representing all.

$$f_t = \sigma(w_f * [x_t, h_{t-1}] + b_f) \quad (3.1)$$

where $\sigma(x) = \frac{e^x}{1+e^x}$, w_f is the weights associated with the forget layer and b_f is the bias.

The second layer is called the “input layer” where the cell state is updated based on the current inputs. Another sigmoid neuron is used to compute a number i_t that decides the amount the current input affects the cell state. This i_t is then multiplied by the input information normalized by the hyperbolic tangent function \hat{C}_t .

$$i_t = \sigma(w_i * [x_t, h_{t-1}] + b_i) \quad (3.2)$$

$$\hat{C}_t = \tanh(w_c * [x_t, h_{t-1}] + b_c) \quad (3.3)$$

where $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ and w_* and b_* are the weights and biases of the respective layers. With these two layers defined, the cell state, C_t can now be calculated for time t as,

$$C_t = i_t * \hat{C}_t + f_t * C_{t-1} \quad (3.4)$$

The final layer known as the “output layer” where the output for the current time is computed using the cell. However, before the output is calculated, the parts of cell state to use is calculate by another sigmoid layer. This is represented by o_t and can be calculated as,

$$o_t = \sigma(w_o * [x_t, h_{t-1}] + b_o) \quad (3.5)$$

Finally the output is given by,

$$h_t = o_t * \tanh(C_t) \quad (3.6)$$

This process is repeated recursively for all time steps for a given training data and the optimized weights and biases are calculated as for a regular neural network.

3.1.2 Reinforcement Learning

Reinforcement learning [7], [56] is a concept generally associated with psychology for instance, in training of pets by giving them rewards for good actions that the pets performed and penalizing for the bad ones. In time, the pets learn to react to certain situations to maximize the rewards. This concept of modifying behaviors over time by providing rewards can be extended to autonomous agents that need to make decisions given a certain state of the environment.



Figure 3.4: Example of a grid world. The numbers on each block represent the reward for the block. The goal is to move in a manner to maximize the reward and reach the goal.

In robotics application reinforcement learning is used extensively to model hard-to-engineer behaviors using rewards/penalizing. It can be used to control for under actuated robots like acrobats and cart-pole [57] and quadrotors [58] and also for very complex systems like humanoids [59] and self driving cars [60]. A more detailed survey of the use of reinforcement learning is done by Kober et. al. [61].

Before moving onto the mathematics of reinforcement learning, consider the figure shown in Fig. 3.4. This figure shows a grid world where an agent starts from the start grid and has to move to the goal grid. The agent cannot pass through the wall and the numbers shown on the grid are the rewards received by the agent when it comes to the grid. The agent does not know the objective of the game before the start (to reach the goal) but only knows to maximize the reward. From the start position the agent will start exploring by moving to its neighboring grids and over time with experience in the grid world it will find a most efficient path to

the goal. This example describes the reinforcement learning in a crude way and is presented here to help readers understand the importance of reinforcement learning. The power of reinforcement learning is that it enables to code in complex behaviors using simple reward functions. In this particular example, the final path that the agent would decide would include initially going to the block with +1 reward before going to the goal. Similarly if it was desired that the agent go to a another block before going to the goal, that block could be given a positive reward. Now, lets move on to a more mathematical formulation of the algorithm.

Given an environment and an agent that needs to interact autonomously with the environment, for each action taken by the agent, a new (or same) state of the environment will be reached and the environment will provide a reward according to the desired goal of the problem. The objective of reinforcement learning is for the agent to perform actions based on the state of the environment such that it can collect maximum possible rewards over time. Here, the agent needs to use its “experience” in order to know which actions might result in greater rewards in the future. Immediate rewards matter less than the expected rewards while choosing these actions. Also, the action to be chosen must be based only on the knowledge of the current state and not the previous states. This property is known as Markov property and such a decision process is known as Markov Decision Process [62] (MDP). Formally, an MDP can be defined using the terms below:

S : A set of all possible states, s , of the environment, called as State Space.

A : A set of all possible actions, a , that an agent can take, known as Action Space.

$T(s'|s, a)$: Transition Probability Matrix, mapping probability of reaching state s' given the environment was in state s and an action a was taken by the agent.

$R(s, a, s')$: Reward as a function for a particular action a taken in state s and

reaching state s' .

γ : Discount factor denoting how important current reward is with respect to previous reward.

Given a MDP, the reinforcement learning problem is to maximize the expected rewards at each time and perform an action accordingly. That is to calculate a policy, the probability of taking an action a given a state s , denoted by $\pi(a|s)$ for all actions $a \in A$ such that it maximizes total rewards over time.

$\pi(a|s)$: Probability of taking an action a , given a state s .

To solve this problem we define two functions value $V(s)$ and action-value $Q(s, a)$ which define value of a state and value of a state-action pair respectively. These are defined below:

$V(s)$: Expected total rewards received by visiting this state.

$Q(s, a)$: Expected total rewards received by visiting this state and performing an action a .

These functions are evaluated using the Bellman equations for a given policy π :

$$\begin{aligned}
 V_\pi(s) &= \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s \right] \\
 &= \mathbb{E} \left[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \middle| S_t = s \right] \\
 &= \sum_a \pi(a|s) \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V_\pi(s') \right]
 \end{aligned} \tag{3.7}$$

Similarly for action-value function,

$$\begin{aligned}
 Q_\pi(s, a) &= \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a \right] \\
 &= \mathbb{E} \left[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \middle| S_t = s, A_t = a \right] \\
 &= \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V_\pi(s') \right]
 \end{aligned} \tag{3.8}$$

And correspondingly policy can be update as,

$$\pi(a|s) = \frac{e^{\beta Q(s,a)}}{\sum_a e^{\beta Q(s,a)}} \tag{3.9}$$

This is called a softmax or Boltzmann action selection where β is a constant. This makes the policy a differentiable function which is necessary if it is to be used for inverse reinforcement learning.

There are several different algorithms like Monte-Carlo and SARSA that can be used to calculate the value functions and action-value functions and in turn calculate the policy. In this thesis, a dynamic programming algorithm was used to calculate these functions. The algorithm is described as below (assuming R is only a function of current state):

Algorithm 1: Reinforcement Learning Algorithm

Input: $\text{MDP}(S, A, T, R, \gamma)$

```

1  $V(s) \leftarrow R(s);$ 
2 for 1 to  $N$  do
3    $Q(s, a) \leftarrow R(s) + \gamma \sum_{s'} T(s, a, s') V_\pi(s');$ 
4    $\pi(a|s) \leftarrow \frac{e^{\beta Q(s,a)}}{\sum_a e^{\beta Q(s,a)}};$ 
5    $V(s) \leftarrow \sum_a \pi(a|s) Q(s, a);$ 
6 end
```

Output: Policy, $\pi(a|s)$

3.1.3 Inverse Reinforcement Learning

For learning a response to a certain situation, it is not always possible to find a suitable reward function that can describe the objective to be achieved. Sometimes it is better to learn from a demonstration and infer the underlying rewards and act according to those. For analogy, we ourselves learn most of the things by watching other people around us, for instance, language. We infer from other people are speaking and infer what words are good and bad to use in some situations. Yes, there are some times where corrections happen by teachers and parents but major part of learning happens by observing others speak. A similar approach to learning actions for an autonomous agents using “expert’s” data is known as inverse reinforcement learning or learning from demonstrations. This concept was introduced by Ng. et. al. [38]. Inverse reinforcement learning utilizes the framework of Markov Decision Processes without a reward function but with addition of an expert’s data. The reward functions are inferred from this expert’s data and reinforcement learning is used to calculate the policy.

However, problem statement of inverse reinforcement learning has been shown to be an ill-posed problem [38]. That is the given expert’s data is not enough to define a unique reward function for the state. To solve this problem, a reward function is represented as a linear combination of a set of feature basis functions. A feature basis function for a reward over state, $R(S)$ would be a function $\phi : S \rightarrow \mathbb{R}$. Hence, reward can be written as ,

$$R_{\mathbf{w}}(s) = \sum_i w_i \phi_i(s) \quad (3.10)$$

where $\mathbf{w} = [w_0, w_1, w_2, \dots]^T$, $\phi = [\phi_0, \phi_1, \phi_2, \dots]^T$ are the feature weights and feature vectors respectively. The choice of the feature vectors thus have a great

impact on the behaviors obtained. These require some intuition and the knowledge of the desired behaviors at a high level. It is important to know what features can affect the behavior even if it is not known how exactly they affect it. Moreover, the more the features the better it is for the algorithm.

Calculating the optimal weights now becomes the objective of inverse reinforcement learning. These weights are calculated based on maximizing the likelihood of the given expert trajectories which are set of state-action pairs, $\tau = [s_0, a_0, s_1, a_1, s_2, a_2, \dots]$ and $D = [\tau_1, \tau_2, \tau_3, \dots]$. Since a trajectory is made up of particular states, the reward of a trajectory is defined as,

$$R_{\mathbf{w}}(\tau) = \sum_{s \in \tau} R_{\mathbf{w}}(s) \quad (3.11)$$

The thesis uses the algorithm of maximizing the likelihood of the given expert's trajectories over the feature weights. This algorithm was introduced in the paper by Ziebart et.al. [44]. This paper makes an assumption that probability of a trajectory is directly proportional to the exponent of the reward of the trajectory, i.e.,

$$P(\tau|\mathbf{w}) \propto e^{R_{\mathbf{w}}(\tau)} \quad (3.12)$$

And the normalizing constant Z also known as partition function is the sum of probability of all possible trajectories in the state space,

$$Z(\mathbf{w}) = \sum_{\forall \tau} e^{R_{\mathbf{w}}(\tau)} \quad (3.13)$$

The objective of inverse reinforcement learning can now be defined as maximizing the log likelihood of the expert's trajectories, $\max L_{\mathbf{w}}$, where $L_{\mathbf{w}}$ is,

$$\begin{aligned}
 L_{\mathbf{w}} &= \frac{1}{|D|} \ln \left[\prod_{\tau \in D} P(\tau|\mathbf{w}) \right] \\
 &= \frac{1}{|D|} \sum_{\tau \in D} \ln P(\tau|\mathbf{w}) \\
 &= \frac{1}{|D|} \sum_{\tau \in D} \left[R_{\mathbf{w}}(\tau) - \ln Z(\mathbf{w}) \right] \\
 &= \frac{1}{|D|} \sum_{\tau \in D} R_{\mathbf{w}}(\tau) - \ln Z(\mathbf{w}) \\
 &= \frac{1}{|D|} \sum_{\tau \in D} R_{\mathbf{w}}(\tau) - \ln \sum_{\forall \tau} e^{R_{\mathbf{w}}(\tau)}
 \end{aligned} \tag{3.14}$$

The optimal weight can be defined as,

$$\mathbf{w}^* = \operatorname{argmax}_{\mathbf{w}} \frac{1}{|D|} \sum_{\tau \in D} R_{\mathbf{w}}(\tau) - \ln \sum_{\forall \tau} e^{R_{\mathbf{w}}(\tau)} \tag{3.15}$$

To maximize, the gradient of L_w with respect to w ,

$$\begin{aligned}
 \nabla_w L &= \frac{1}{|D|} \sum_{\tau \in D} \frac{dR_w(\tau)}{dw} - \frac{1}{\sum_{\forall \tau} e^{R_w(\tau)}} \sum_{\forall \tau} e^{R_w(\tau)} \frac{dR_w(\tau)}{dw} \\
 &= \frac{1}{|D|} \sum_{\tau \in D} \phi_{\tau} - \sum_{\forall \tau} \frac{e^{R_w(\tau)}}{\sum_{\forall \tau} e^{R_w(\tau)}} \phi_{\tau} \\
 &= \frac{1}{|D|} \sum_{\tau \in D} \phi_{\tau} - \sum_{\forall \tau} P(\tau|w) \phi_{\tau}
 \end{aligned} \tag{3.16}$$

where $\phi_{\tau} = \sum_{s \in \tau} \phi(s)$

Since, all trajectories possible in the state space is not possible to compute, the second term in the above expression can be approximated to $\sum_{s \in S} P(s|w) \phi(s)$. This can be written because the trajectories are made up of states and therefore a sum over all trajectories will be equivalent to sum over all states. Hence, the gradient

now is,

$$\nabla_w L = \frac{1}{|D|} \sum_{\tau \in D} \phi_\tau - \sum_{s \in S} P(s|w) \phi(s) \quad (3.17)$$

where the term $P(s|w)$ is known as the state visitation frequency (SVF) for state $s \in S$. This can be defined as the probability of visiting a state s at any given time for the particular weights (or policy). To calculate the SVF, a forward pass of the reinforcement learning with the current set of random weights is used and the optimal policy $\pi_{\mathbf{w}}(a|s)$ is calculated. Also, an initial estimate of probability distribution, $\mu_0(s)$ of visiting a state s is calculated using the trajectories. Further this probability is update over time t by the equation,

$$\mu_{t+1}(s) = \sum_a \sum_{s'} \pi_{\mathbf{w}}(a|s) T(s'|s, a) \mu_t(s') \quad (3.18)$$

And finally, the SVF is given by,

$$P(s|\mathbf{w}) = \sum_t \mu_t(s) \quad (3.19)$$

This can now be substituted back in the equation 3.17 to calculate the gradient and the weights can be updated as,

$$\mathbf{w} = \mathbf{w} + \eta \nabla_{\mathbf{w}} L \quad (3.20)$$

where η is a constant termed as learning and determines how fast the weights converge to the maxima. Though too large a value of the learning rate implies that it weights might keep oscillating around the maxima but never converge to it and too small a value might get the objective function stuck in a local minima, if it is not too slow to converge. Hence, a proper tuning for this parameter needs to be

done for good results.

Furthermore, once the algorithm converges or number of iterations are done, a vector of weights is obtained along with the optimal policy corresponding to the weight vectors.

Algorithm 2 below summarizes the section here and presents the inverse reinforcement learning algorithm used during the course of this thesis.

Algorithm 2: Inverse Reinforcement Learning Algorithm

Input: MDP(S, A, T, γ), expert trajectories $D = [\tau_0, \tau_1, \tau_2, \dots]$, feature vectors: $\phi = [\phi_0, \phi_1, \phi_2, \dots]^T$

- 1 Randomly Initialize $\mathbf{w} = [w_0, w_1, w_2, \dots]^T$;
- 2 **for** 1 to N **do**
- 3 $R_{\mathbf{w}}(s) \leftarrow \sum_i w_i \phi_i(s)$;
- 4 Calculate $\pi_{\mathbf{w}}(a|s)$ from Algorithm 1 using $R_{\mathbf{w}}(s)$;
- 5 **for** $t \leftarrow 0$ to T **do**
- 6 $\mu_{t+1}(s) \leftarrow \sum_a \sum_{s'} \pi_{\mathbf{w}}(a|s) T(s'|s, a) \mu_t(s')$;
- 7 **end**
- 8 $P(s|\mathbf{w}) \leftarrow \sum_t \mu_t(s)$;
- 9 $\nabla_w L \leftarrow \frac{1}{|D|} \sum_{\tau \in D} \phi_{\tau} - \sum_{s \in S} P(s|w) \phi(s)$;
- 10 $\mathbf{w} \leftarrow \mathbf{w} + \eta \nabla_{\mathbf{w}} L$
- 11 **end**

Output: Weights: \mathbf{w} , Policy: $\pi_{\mathbf{w}}(a|s)$

3.2 Methodology

In this chapter, neural networks, reinforcement learning and inverse reinforcement learning algorithms were introduced. This section ties in all of these concepts to provide an integrated view of how the endoscopic camera is to be automated. Fig. 3.5 shows the flowchart for the same.

The inverse reinforcement learning algorithm outputs a policy which specifies actions given a particular state of the environment. Since the behavior of camera should be different for different intention of the user operating master tool manip-

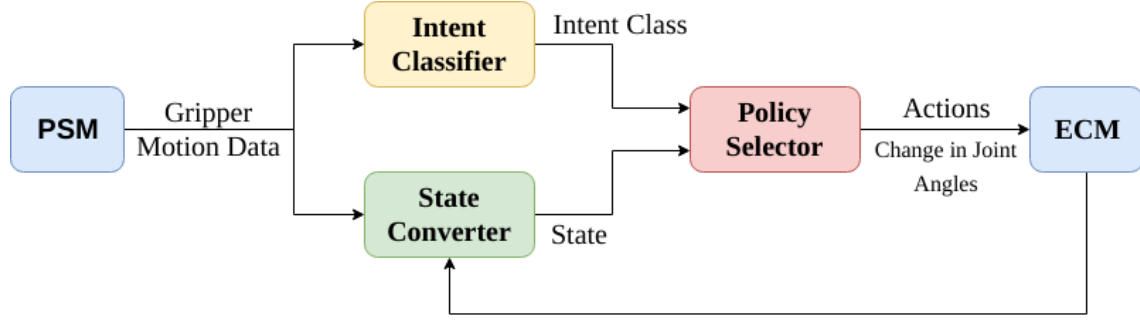


Figure 3.5: Flowchart showing how the automation of endoscopic camera manipulator (ECM) based on the master tool manipulator (MTM) motion

ulator, the policy for automating camera motion should be different too. For this, trajectories for different classes were segmented out and different policies were obtained for each of the classes of intents. The classification of the master motion is done by the neural networks and the output is used to select the policy that should be used to move the endoscope.

The action output of the policy which is defined to be the change in joint angles of the ECM is a function of state of the environment. This state is calculated based on the current joint angles of the ECM and the gripper pose and its motion.

Detailed calculations to get the states are shown in Chapter 6. The chapter also discusses the transition of states given actions and analyses the trajectories data to obtain the initial state visitation frequency required in Algorithm 2.

Chapter 4

Simulation Environment

This chapter describes the current system architecture of the daVinci research kit and the current simulations available for the teleoperated system. New simulation environment and models were developed during the course this thesis using open source dynamic simulators such as Gazebo [69] and open source set of libraries CHAI3D that were mainly developed for haptic interfaces. However, due to some issues regarding stability and time delay, simulation using Gazebo simulator was not used for the thesis. The CHAI3D simulator was used and interfaced with a virtual reality headset Oculus Rift. A user study was conducted using this simulation environment (described in next chapter) for collecting data on how endoscope should be moved with respect to the instruments.

The chapter also introduces the kinematics of the Endoscopic Camera Manipulator (ECM). This calculations are required in further calculations for the inverse reinforcement learning algorithm.

4.1 daVinci Research Kit

The daVinci Research Kit (dVRK) [63] consists of manipulator arms from retired clinical daVinci surgical systems donated by Intuitive Surgical to universities and research groups to expand the areas of research in the field. The research kit provided by Intuitive included limited hardware and software capabilities due to intellectual property rights of the company. A collaboration between Worcester Polytechnic Institute (WPI) and Johns Hopkins University (JHU) provided the controllers for establishing hardware interface between each component of the dVRK [63]. The controller boards consist of Quad-linear Amplifiers with FPGAs handling joint control and sensor data.

These data were integrated with a core set of libraries called “Computer Integrated Surgical Systems and Technology” (CISST) which provide basic libraries for math, multi-threading and data logging [64]. These libraries were utilized by the surgical assistant workstation (SAW) system architecture to provide an API for general medical robots using teleoperation [65]. These handle the higher level implementation such as forward, inverse kinematics, joint controllers and cartesian controllers.

An additional layer of interface was added to this system architecture by Chen et. al. [67] (Fig. 4.1) using robot operating system (ROS) [68] for communication between a linux computer and the hardware. This made the interface to use the daVinci Research Kit easier to use, removing the need to understand the CISST-SAW libraries. Additionally it provided a way to visualize the hardware and its state using the visualization tool RViz provided by ROS and allowed developments in the open source community like sensor modules to be used with dVRK.

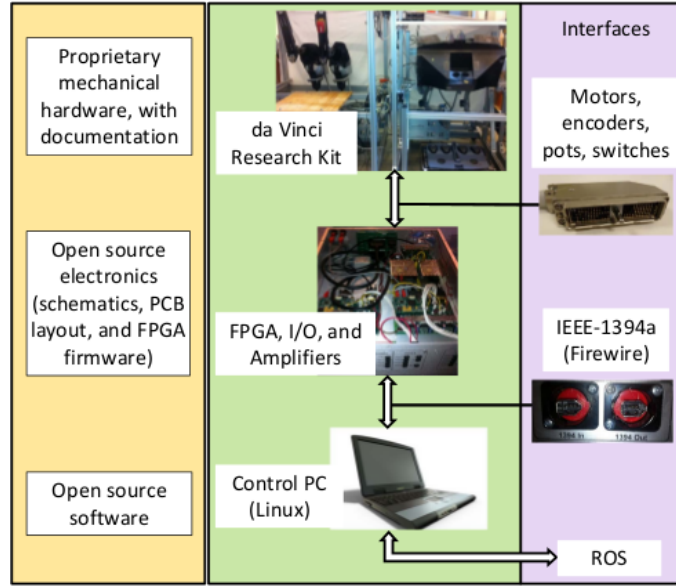


Figure 4.1: Overview of the daVinci research kit including the hardware provided by Intuitive Surgical, and open source electronics and software layers as detailed in the work [67]

4.1.1 Gazebo Simulations

Though RViz is a very good tool to be used along with the hardware to visualize the state of the manipulators, it does not have any abilities to create a simulation itself. There is no physics engine to simulate the dynamics or create an environment to have simulated cameras which capture images from the simulated endoscopic camera. For this, Gazebo [69] was used in the initial stages of the thesis. It provided a way to simulate all the manipulators and dynamics independent of the hardware and would allow stereo images to be captured. Moreover, different environments could be created and interacted with using the manipulators.

For the simulation environment, new models were created which followed a closed loop kinematics of the hardware rather than the simplified serial chain models previously used. This work was done in collaboration with a colleague, Radian Azhar. Fig. 4.2 shows these newer models simulated in the gazebo environment. The simu-

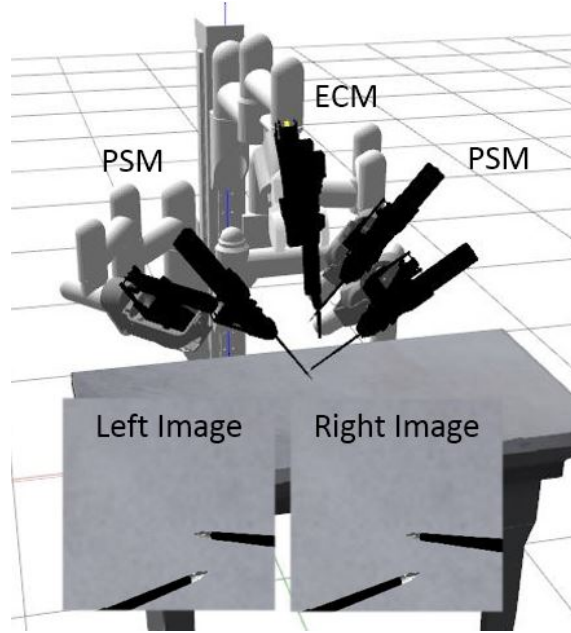


Figure 4.2: The whole daVinci Research Kit simulated in Gazebo simulator. A stereo image from the simulated ECM has been shown along with the models.

lator allowed creation of stereo images of the endoscope which is essential for leaning how to automate the endoscope.

However, the simulations thus created were not stable enough. Since the dynamic parameters were not exactly known and were only estimated from Solidworks, the models vibrated in certain configurations. Furthermore, there was a high parity in the simulation and real time because of the calculations for dynamics of the closed loop system, so it was difficult to control the manipulators using external haptic devices and Oculus Rift. Therefore this environment was not usable for the thesis. Hence, a different environment as described below was chosen.

Before describing the simulation environment, the forward and inverse kinematics of the ECM is calculated.

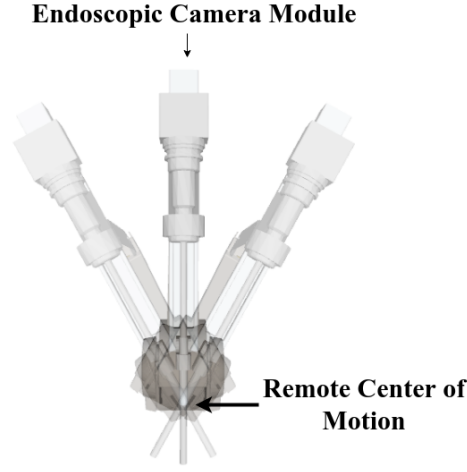


Figure 4.3: Figure showing the remote center of motion (rcm) mechanism for the Endoscopic Camera Manipulator (ECM).

4.2 Endoscope Kinematics

This section provides the forward and inverse kinematics of the ECM. All the manipulators that operate on the patient side of the daVinci surgical system including the ECM have a remote center of motion (rcm) mechanism. This center of motion is fixed in space and does not move with the change in joint angles of the manipulator. Due to this feature, the cartesian coordinates of the end effector of the manipulator can be written with respect to the rcm and need not be written with respect to the base of the manipulator. Fig. 4.3 shows the rcm for different configurations of the ECM.

4.2.1 Forward Kinematics

Endoscopic camera manipulator is a 4 degree of freedom manipulator with a remote center of motion mechanism. The manipulator is designed in such a way that a particular point, called the remote center of motion (RCM) is always fixed in space. To achieve this, a parallelogram mechanism had been designed. Due to the remote

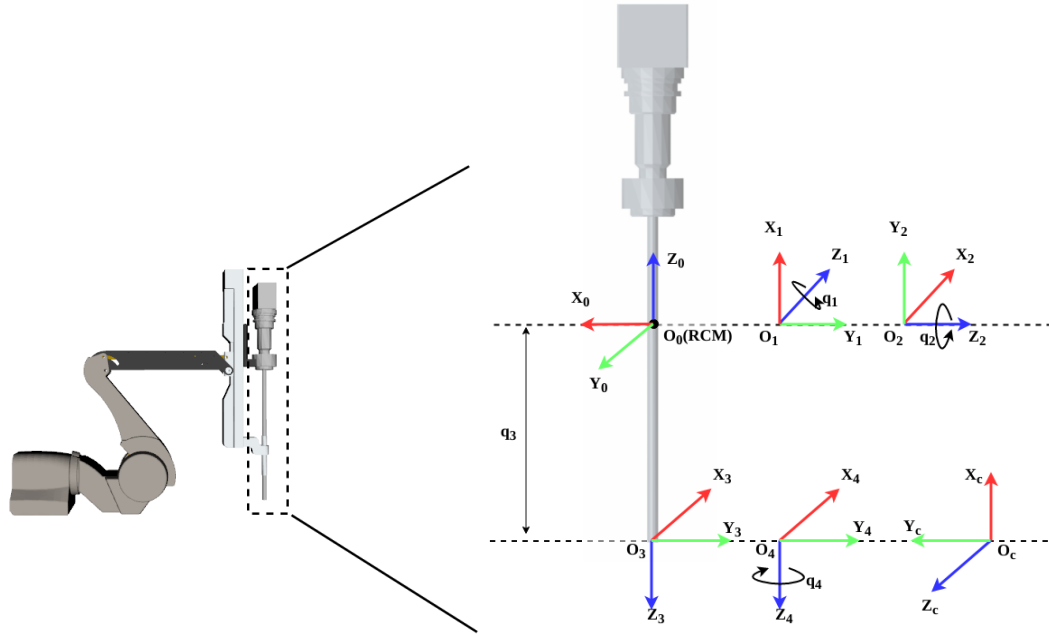


Figure 4.4: Figure showing frame assignment for forward kinematics of ECM used to calculate the modified D-H parameters specified in Table 4.1. The dotted lines represent that the frames on these lines are at the same origin and are not translated horizontally. These frames can be plotted with respect to the rcm instead of the base of ECM.

center of motion, the kinematics of the manipulator is simplified and can be written with respect to the RCM. Fig. 4.4 shows the frame designation according to the modified Denavit-Hartenberg convention for deriving the transformation between two frames. In the figure, the dotted horizontal lines represent that the frames along the dotted line are at the same position on the endoscope. They are not translated horizontally. The corresponding parameters for each of the frames are given in the Table 4.1 along with the type of the joints.

Table 4.1: Modified Denavit-Hartenberg parameters for the endoscopic camera manipulator (ECM).

| Joint | Joint Type | α | a | θ | d |
|---------|------------|------------------|-----|-----------------------|-------|
| Joint 1 | Revolute | $\frac{\pi}{2}$ | 0 | $q_1 + \frac{\pi}{2}$ | 0 |
| Joint 2 | Revolute | $-\frac{\pi}{2}$ | 0 | $q_2 - \frac{\pi}{2}$ | 0 |
| Joint 3 | Prismatic | $\frac{\pi}{2}$ | 0 | 0 | q_3 |
| Joint 4 | Revolute | 0 | 0 | q_4 | 0 |

Using these parameters and the modified D-H convention to get the transformation matrix between the frames, T_4^{rcm} (frame 4 with respect to the RCM) can be calculated. Further, there is a fixed transformation matrix from the Frame 4 to the camera frame given by T_c^4 where,

$$T_c^4 = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.1)$$

And $T_c^{rcm} = T_4^{rcm} * T_c^4$. This completes the forward kinematics for the endoscope.

4.2.2 Inverse Kinematics

Inverse kinematics of a manipulator arm is the calculation of joint angles given a particular pose of the end effector of the manipulator arm. This is essential when working in the joint space of a manipulator arms.

Given a transformation matrix,

$$T_c^{rcm} = \begin{bmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ t_{41} & t_{42} & t_{43} & t_{44} \end{bmatrix}$$

that specifies the camera frame with respect to the RCM, the joint angles can be calculated using the equations:

$$q_2 = \arctan2 \left(t_{21}, \sqrt{t_{11}^2 + t_{31}^2} \right) \quad (4.2)$$

$$q_1 = \arctan2 \left(-\frac{t_{11}}{\cos(q_2)}, -\frac{t_{31}}{\cos(q_2)} \right) \quad (4.3)$$

$$q_4 = \arctan2 \left(-\frac{t_{22}}{\cos(q_2)}, -\frac{t_{23}}{\cos(q_2)} \right) \quad (4.4)$$

$$q_3 = \sqrt{t_{14}^2 + t_{24}^2 + t_{34}^2} \quad (4.5)$$

4.3 Chai3D environment

In this thesis, a simulation environment is necessary as current hardware systems do not provide the abilities to move endoscope camera while the instruments are

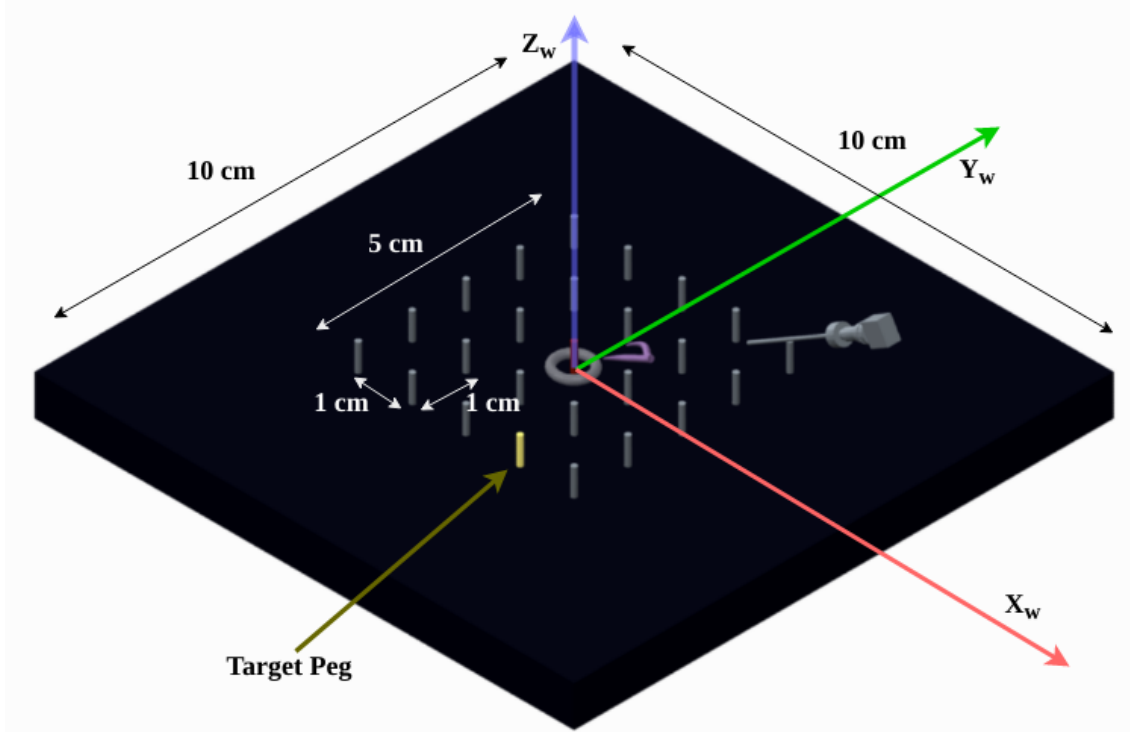


Figure 4.5: Isometric View of the Simulation Environment

moved. Moreover, it is too risky to try new control algorithms generated using stochastic calculations like reinforcement learning on the expensive hardware before validating on a simulation environment.

Here a set of open source C++ libraries called CHAI3D [73], available online at [74], are used. These libraries provide a haptic and dynamic simulation capacities and has modules developed using BULLET [75] as the physics engine for dynamic interaction between objects. The visualization and rendering are done using the open graphics library OPENGL. In addition, various haptic devices can be interfaced and used with these set of libraries to create an interactive simulation environment.

At the AIM lab, these libraries were modified to have the MTM interfaced with the chai3d libraries by a colleague Adnan Munawar, and the work is available [76]. Furthermore, a simulation environment for haptic force feedback and a layout for integration with ROS was done.

Using these libraries, I created a framework which simulates the ECM kinematics along with stereo images in the chai simulation environment and wrote interfaces to control the endoscope using the Oculus Rift. Furthermore, an environment was created to perform a pick and place task using a gripper as shown in Fig. 4.6 and 4.5. The task is to pick up the ring from the current position and place it on the peg which is blinking yellow. As soon as the ring is placed, the peg turns red and another peg would turn yellow. This task can be performed while wearing the Oculus and the environment would change with the head movements. The calculations for these have been shown in the next section. The endoscope shown in these figures moves with the remote center of motion and with the same kinematics as the hardware.

As has been mentioned in Chapter 1, this particular task of pick and place was chosen from the five basic task that are used for training surgeons according to [5] because it was thought that this task required the most use of the camera since the instruments are always moving and would be moving out of screen. Other tasks such as suturing, knot tying, needle passing are more constrained to a local workspace and would require the camera to be still rather than moving. Hence, the pick and place was chosen to demonstrate the approach using inverse reinforcement learning to automate the endoscope camera motion.

The three out of the four degrees of freedom (roll, pitch and yaw) of the simulated ECM were controlled using the Oculus Rift. The fourth degree of freedom (third joint), insertion was controlled using the foot pedal tray, more specifically the +/- pedal of the tray was directly mapped to control the third joint angle and allow zoom in and zoom out of the scene.

The environment created is a 10cm x 10cm plane with 25 pegs arranged in a 5x5 grid with each peg 1cm apart. The gripper used is controlled using the MTM of the daVinci system. The gripper is only a representation of teleoperation and

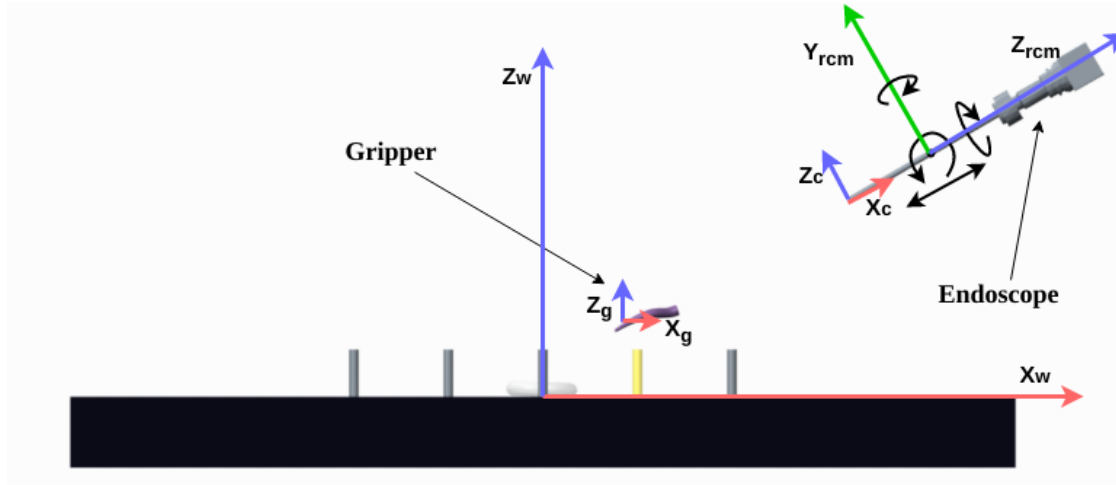


Figure 4.6: Side View of the Simulation Environment showing the gripper, ring, peg board and the endoscope.

does not follow the kinematics of the PSMs. It was created in order to simplify the simulation. Though the method presented in the thesis can be applied to the instruments of the PSM as well and can be extended to various tasks.

The position of the all objects in the simulation are defined with respect to the world frame (denoted by subscript “w”) shown in Fig. 4.6 along with the location of the RCM (denoted by subscript “rcm”) of the ECM, the degrees of freedom of the endoscope and the camera frame (denoted by subscript “c”). The RCM frame is fixed throughout the whole task and the transformation is given by T_{rcm}^w ,

$$T_{rcm}^w = \begin{bmatrix} 0 & -\sin \frac{\pi}{3} & \cos \frac{\pi}{3} & 0.03 + 0.05 * \cos \frac{\pi}{3} \\ 1 & 0 & 0 & 0 \\ 0 & \cos \frac{\pi}{3} & \sin \frac{\pi}{3} & 0.06 + 0.05 * \sin \frac{\pi}{3} \end{bmatrix} \quad (4.6)$$

This was set so that the initial joint angles of $q = [0, 0, 0.05, 0]$ would be looking at the center of the world from an angle of 60° . The forward kinematics of the ECM can now be used to find the transformation between the world frame and the camera frame, $T_c^w = T_{rcm}^w * T_c^{rcm}$. This transformation is required to get the states at any

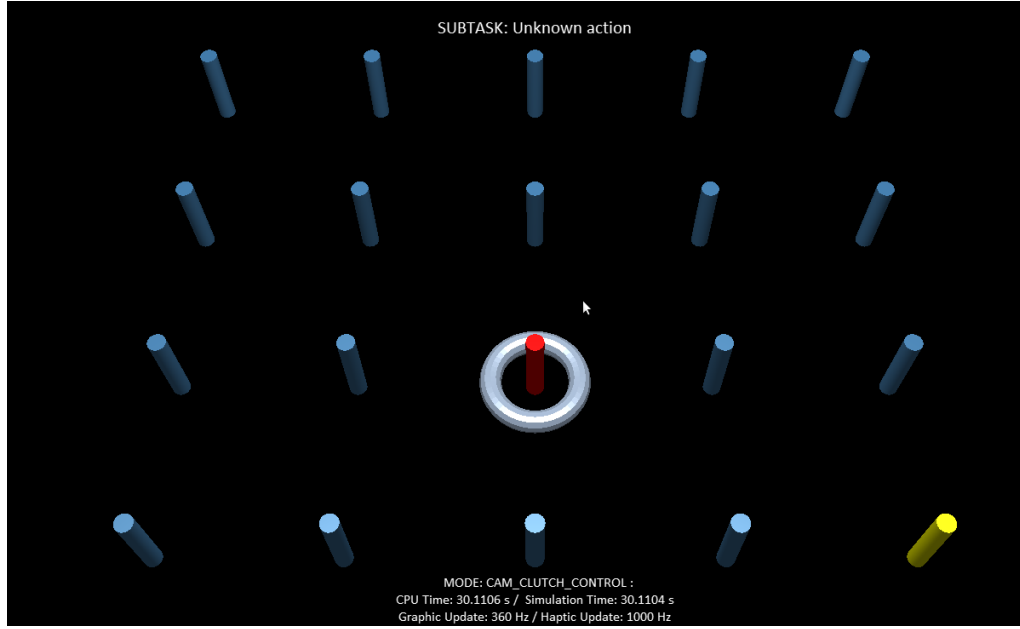


Figure 4.7: Point of view of the left camera of the stereo camera on the endoscope at the initial joint configuration of the ECM showing the ring and the target peg in yellow at bottom right. If the target peg is not visible, a translucent yellow arrow pops up pointing towards the target peg.

Table 4.2: Table showing different parameters of the stereo camera in the simulation

| Parameters | Value |
|---------------------|------------|
| Field Of View | 30° |
| Stereo Separation | $0.01m$ |
| Focal Length | $2m$ |
| Near clipping Plane | $0.001m$ |
| Far clipping Plane | $1m$ |

particular time for the reinforcement learning algorithm as described in Chapter 6.

Fig. 4.7 shows the point of view of the stereo camera (left camera) in the simulated environment. This stereo image is sent to the Oculus Rift and the powerful lenses in the headset create a 3D environment. The different parameters of the simulated stereo cameras are presented in Table 4.2.

4.4 Oculus Rift Setup

Oculus Rift CV1 is a virtual reality headset developed by Oculus VR, Menlo Park, California, 2018. In the headset there are two very powerful lenses that create a wide field of view and when a stereo image is displayed on the lenses, it gives an immersive 3 dimensional view of the system. For this thesis, the consumer version of the rift, Oculus Rift CV1, Oculus VR, Menlo Park, California, 2018 was used. Although, this version is released to be used only on a Windows operating system, open source libraries to interface the headset with an ubuntu system are available and in particular the OpenHMD repository [70] was used. A stereo image of the simulation from the simulated endoscope as described in the next section was sent to the headset to provide a 3D virtual reality game for picking and placing an object. This enabled the data from the headset to be communicated using ROS.

In addition to the lenses, the headset has an accelerometer, gyro and a magnetometer [72] sensor which work in combination to give the orientation of the headset at any given time. The open source libraries give this orientation as a quaternion representation. An additional open source library called the `openhmd_ros` [71] was used to provide the ROS bindings for the OpenHMD. This allowed the rift to be used with ROS for communication and recording of the data.

The data for the orientation of the headset were then mapped to the endoscope motion. This mapping enables the user of the headset to move their head and the scene of the environment changes accordingly. That is, if the head is moved right, the image in view should also be shifted right. Fig. 4.8 shows the frames of Oculus when worn by a person.

Since the image of the simulation environment described below is generated using a simulated camera, the motion of the head should be mapped to the motion of the

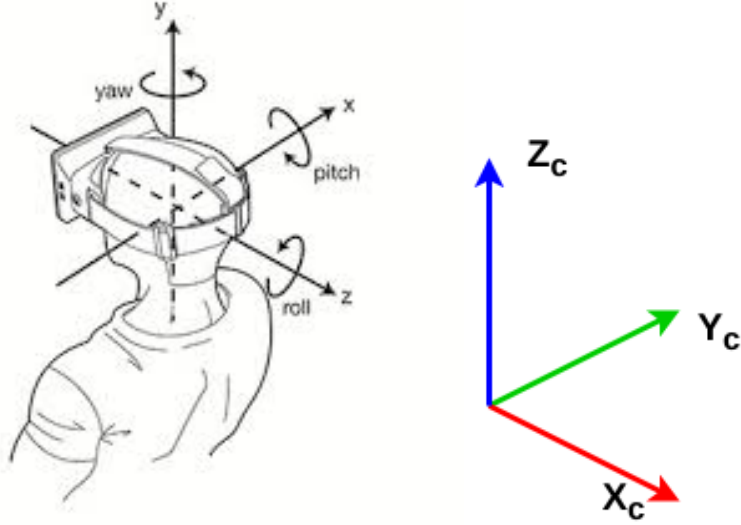


Figure 4.8: (left) Frames of the Oculus Rift as worn by the user with respect to the real world [72]. This frame is tracked using the sensors in the headset with respect to the world. Since the image captured by endoscope is sent to the oculus, the orientation of endoscope with respect to the oculus is shown on the right.

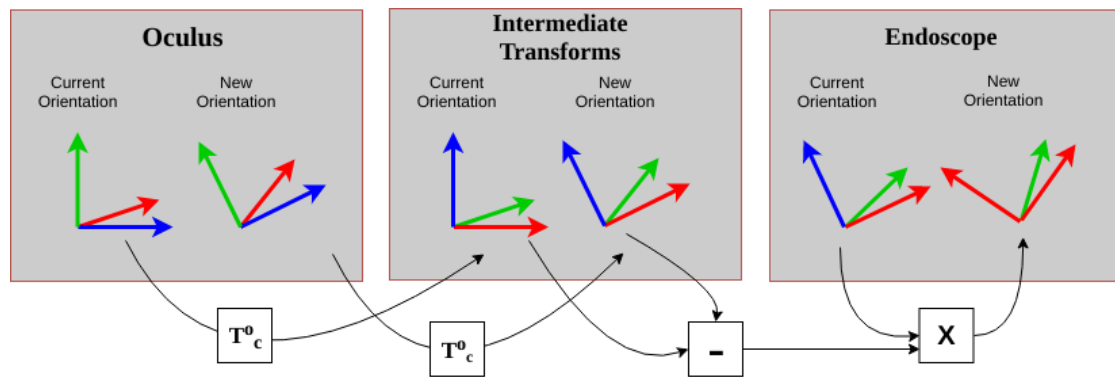


Figure 4.9: Frame transformations showing the change in orientation of the endoscope camera due to the change in orientation of the oculus. The difference between two frames depicted by “-” is actually the change in orientation mathematically given by Eq. 4.7 and multiplications denoted by “x” are post multiplication of the matrices.

endoscope. For this, a change in orientation of the oculus should result in a change in orientation of the endoscope. Fig. 4.9 shows the series of frame transformations to achieve the correct commanded orientation of the endoscope camera based on the head movement of the user.

From this figure, the commanded orientation of the endoscope can be given by the equation,

$$T_{c,t} = T_{c,t-1}(T_{o,t-1}T_c^o)^T(T_{o,t}T_c^o) \quad (4.7)$$

where $T_{c,t}$ is the orientation of the endoscope camera in the simulation world frame, $T_{o,t}$ is the orientation of the oculus in the real world frame and,

$$T_c^o = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \quad (4.8)$$

is the transformation between the oculus and the endoscopic camera as shown in Fig. 4.9.

It should be noted here that the movements of Oculus Rift picked up by head tracking were too sensitive for the endoscope. Therefore a low pass filter was implemented to filter out the jerks and the tremors in the endoscope motion. The headset communicates the data at 50Hz and the joint angles to be commanded are calculated as soon as the data is received using subscriber call back functions. These joint angles calculated using the head tracking information and inverse kinematics as detailed above and were then filtered using the filter

$$q_{t,commanded} = 0.9 * q_{t,calculated} + 0.1 * q_{t-1} \quad (4.9)$$

Chapter 5

Data Collection

The simulation environment described in the previous chapter along with the virtual reality interface achieved with the Oculus Rift was used to collect data by performing a user study. This collected data was used to segment subtasks based on user's motion of the gripper and the head tracking from the virtual reality was used to correlate the movement of the endoscope and the gripper.

This chapter describes the objectives, risks, logistics of the user study performed and the data that was collected in the process. The different segmentation of the pick and place task into subtasks is defined and the procedure to label the data for classification is reported.

5.1 User Study

As mentioned earlier, a user study was performed where participants in the study were asked to perform a pick and place task in a 3D interactive virtual environment using the master of the daVinci surgical system. The user study as approved by the institutional review board and the protocol number is 'IRB-19-0007' and the title of the protocol is 'Automating Endoscopic Camera Motion During Teleoperated Min-

DATA COLLECTION

imally Invasive Surgeries Using Reinforcement Learning’. The study was performed only after the approval from the Institutional Review Board (IRB) was granted. All the guidelines were followed during the course of the study and participants were explained in detail the purpose and potential risks of the study before starting and user’s signed the consent form as per guidelines of the IRB.

During the study, the users were asked to control the gripper in the simulation using the MTM while wearing the headset. The task was to simply pick up the ring using the gripper from its current position and place it on the flashing yellow peg. Once the ring is placed, the peg will turn red and subsequently another peg would start blinking yellow. This task was to be repeated five times so as to give enough data to be collected during this exercise. The pegs would be randomly chosen using a uniform distribution to remove biases in any one direction of motion.

This particular task was chosen for the user study since it was thought that it would be the most intuitive and easy of the tasks to perform for the users. The users for this study were mostly college students from WPI recruited using direct emails and had very limited exposure to the daVinci surgical system and therefore tasks such as suturing, needle passing, etc that require skills and experience in using the system would be difficult to do for a study. Moreover, amongst the basic surgical tasks, the most camera motion is required for a pick and place task where the camera has to be continuously pan and tilt. Other tasks such as knot tying require camera movement before the actual task of knot tying but during the procedure, the camera should be still. Therefore the task for pick and place was chosen for the user study.

The simulation was designed such that the users would have to reposition their heads while performing the task, otherwise it would defeat the purpose of this study. Thus, the field of view was set to a narrow 30° . However, this created a problem in that the user would not know the direction of the target peg if it out of the field of

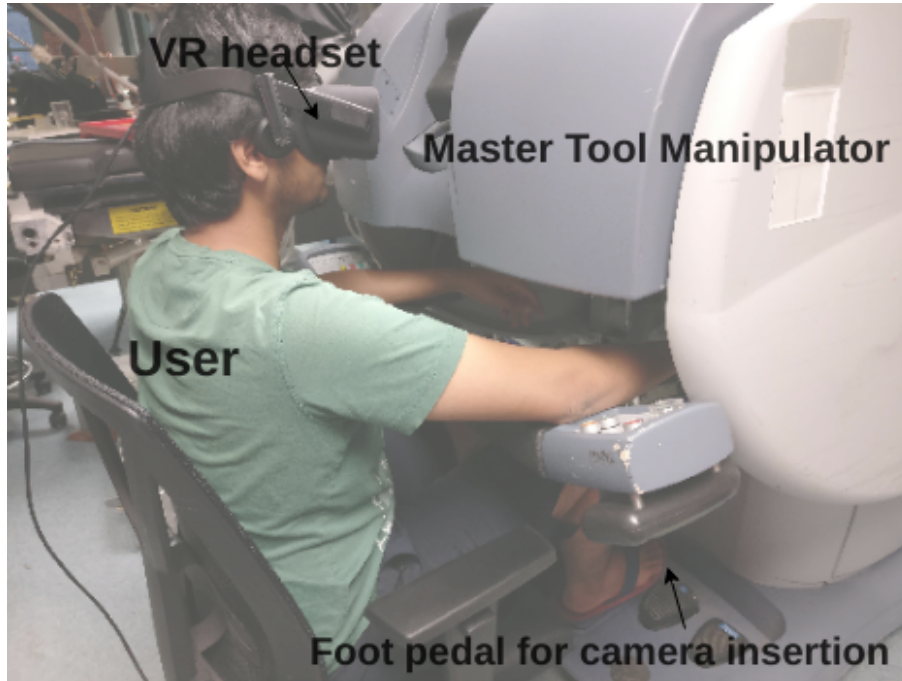


Figure 5.1: A user during the user study completing the exercise using the MTM while wearing the Oculus headset. The simulation environment scene as viewed from the endoscope was streamed to the virtual reality headset.

view and would have to look around to find the peg. This would corrupt the data a bit and these parts would have to be filtered out otherwise the resulting automated motion would capture this motion and hence reward shaky behavior. To avoid this, a widget was created which would immediately pop on the screen if the target peg went out of view. This widget is a thick translucent yellow arrow pointing in the direction of the target peg from the center of the image. This ensured that the user is always aware of where to look and does not wander off the task.

In the user study, 8 participants took part and each participant was asked to complete the exercise 5 times. This resulted in a total 200 pick and place operations to be recorded. The videos of each of these exercises were recorded along with other kinematic data. More details about the data recorded is described in the next section.

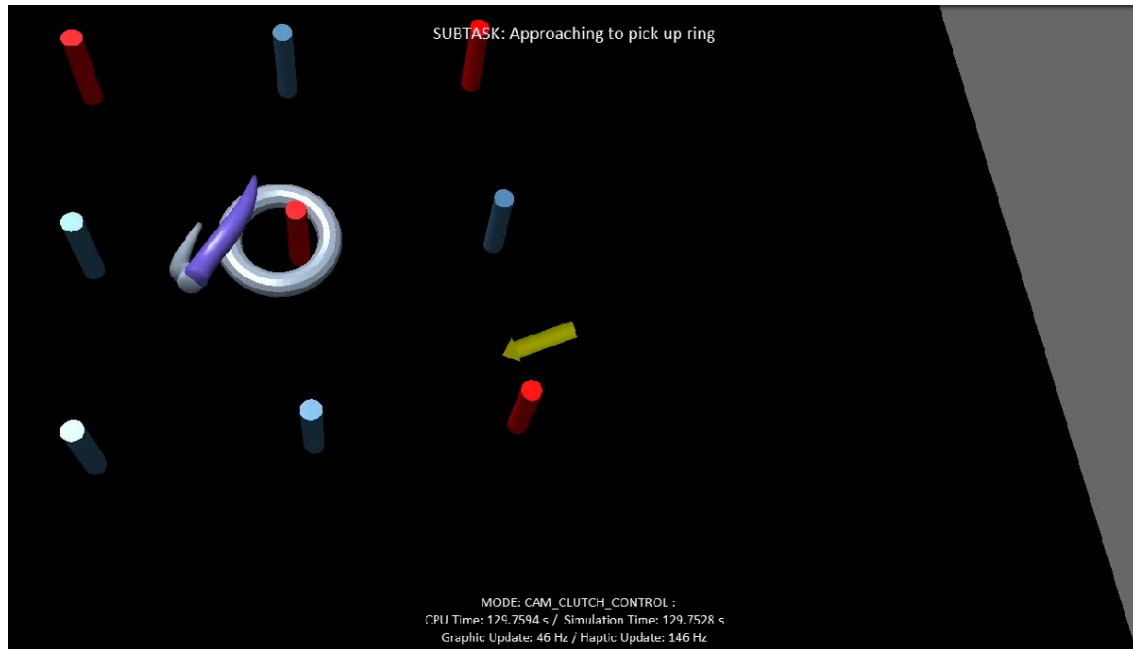


Figure 5.2: Snapshot of the video from one of the user studies showing the arrow pointing towards the target peg whilst the user is approaching to pick up the ring. Additional information regarding the simulation time and frequency of graphic and haptic update is shown at the bottom of the screen and the subtask label for classification is shown at the top.

The participants of the study were all WPI students who volunteered to be a part of this study. Among the 8 participants, 3 were females and 5 were males. All of the participants were recruited by direct mails to each. Average age of the user study group was around 25 years. Most of the participants were very inexperienced with the knowledge and know how of robotic surgery and had never operated the master of the daVinci surgical system. They were allowed sufficient and necessary amounts of trials till they were comfortable and confident enough to record the data. Typically, each study lasted around an hour including the breaks in between exercises to relax. There were no monetary compensations offered to the participants during the study.

The experimental setup of the user study was very straightforward. The MTM was connected to a computer and a ROS core was setup. My laptop running the

simulation along with the headset communicated with the computer and recorded the important data. The users were asked to sit on a chair facing the MTM and grab the MTM handle (right handle for most, left handle for one user was used). Then the headset was placed over the head and the application was started to launch the simulation. Once the simulation booted up, the users were asked if they were comfortable with the starting viewpoint. If they were not, the users were asked to look straight and the viewpoint was reset by changing the endoscope joints to be the initial joint angles. This ensured that the users had the maximum available workspace to move their head around and control the camera movements.

5.1.1 Potential risks during the study

During the study, there were some concerns using the Oculus Rift. These concerns and how they were mitigated have been detailed below:

- The virtual reality headsets can often be too bright and cause a headache if worn for a long time. To ensure the brightness was reduced, the environment background was made using darker colors.
- The powerful lenses of Oculus Rift cause a distortion in the images. This distortion is known as a pin cushion distortion. To counteract the distortion, the image sent to the headset is distorted in an inverse manner (known as Barrel Distortion) to cancel out the effects of the lens distortion as shown in Fig. 5.3. However, after a lot of try it was difficult to achieve a barrel distortion in the CHAI environment. Therefore the participants were specifically asked during the study to not roll their heads and by doing so, the distortion would immediately cause motion sickness.

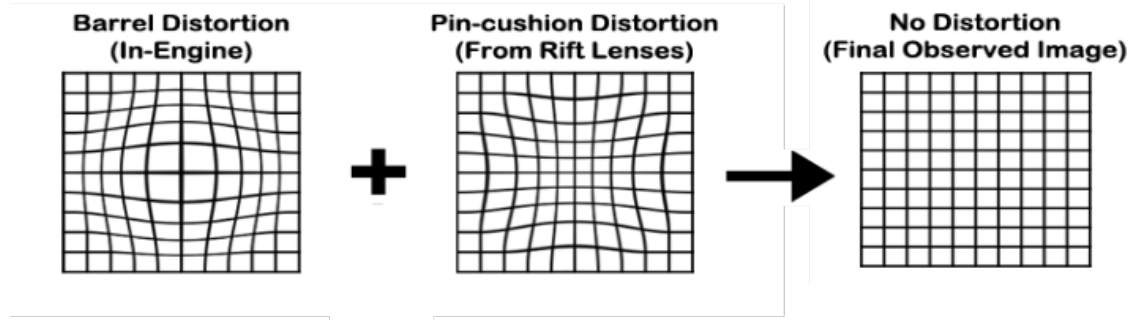


Figure 5.3: Figure showing the pin cushion distortion due to the lenses of Oculus Rift and countering it using barrel distortion.

5.2 Data Collected

The data for the user studies were collected in individual csv files for each exercise. Therefore there were five files recorded for each participant and a total of 225 files were recorded. The recording of data was done using a separate thread that would run through out the exercise and record the relevant data at 1000 Hz. However due to the performance of the computer, the data was recorded at an average frequency of 784.92 Hz with lowest frequency of 723.43 Hz. This is because there are multiple threads (haptics, graphics, running simultaneously with MTM and oculus communicating as well. All the user studies were conducted using a laptop of configuration Intel i7-4510U processor with 4 cores, 8GB RAM and Nvidia GeForce 840M graphics card.

The data collected during the user study were collected for the endoscope motion, gripper motion, the ring position and the target peg position and a label for the subtask at any given time. In total, there were 50 columns of data that was recorded. Table 5.1 gives the description of the recorded data.

Table 5.1: Table describing the variables that were recording during the user study.

| Column Indices | Number of Variables | Description of Variables |
|----------------|---------------------|--|
| 1 | 1 | Real world time calculated using the CPU time. |
| 2 | 1 | Simulated time calculated using timers from CHAI library. |
| 3-6 | 4 | Joint angles of ECM when moved using the head tracking for oculus (j_1, j_2, j_3, j_4). |
| 7-9 | 3 | Position of the camera frame with respect to the world frame in simulation (xyz). |
| 10-18 | 9 | Orientation of the camera frame with respect to the world frame in simulation (R). |
| 19-21 | 3 | Position of the RCM frame of endoscope with respect to the world frame in simulation (xyz). |
| 22-30 | 9 | Orientation of the RCM frame of endoscope with respect to the world frame in simulation (R). |
| 31-33 | 3 | Position of the ring with respect to the world frame in simulation (xyz). |
| 34-36 | 3 | Position of the target peg with respect to the world frame in simulation (xyz). |
| 37 | 1 | Subtask label assigned live when the exercise was done (0 or 1 or 2 or 3 or 4). |
| 38-40 | 3 | Position of the gripper tip with respect to the world frame in the simulation (xyz). |
| 41-49 | 9 | Orientation of the gripper tip with respect to the world frame in the simulation (R). |
| 50 | 1 | Jaw angle of the gripper. |
| 51 | 1 | Subtask relabeled later using frame by frame analysis of the video (0 or 1 or 2 or 3 or 4). |

5.2.1 Video recording

The video of each participant's each trial was recorded for the scene seen through the left camera of the stereo camera. The videos were recorded at 30 fps with screen resolution of 1300x743 in MKV format. The video has time stamps data in addition to the subtask label which was done simultaneous to the user study. This helps in synchronizing the video with the kinematic data collected.

5.2.2 ROS bags

In addition to the kinematic data and the video, data from all communication topics in ROS were also recorded in ROS bags. Using these ROS bags, these can be replayed to create the same simulation including the movements of the gripper and the head movements that were done by the user. This gives an opportunity to compare the automated camera movements with the head tracking data for the same gripper motion. Though ROS bags are useful, they were thought of a bit later during the user studies and therefore were recorded for participants 5 to 9 only.

For the classification of task into subtasks, the next section describes the subtasks in detail and the observations that distinguish each subtask from the other. These segmentation of subtasks have been made keeping in mind the camera movement and is hypothesized that the camera movement would be different for different subtasks.

5.3 Subtask Classes

The task of pick and place a ring onto a peg can be divided into four subtasks, namely:

- Approaching to pick up the ring

- Picking up the ring
- Approaching to place the ring on the peg
- Placing the ring

The task was divided into only these subtasks because these subtasks have different identifying features in terms of the gripper data that help in classifying these subtasks. For instance, approaching to pick up has the jaw open with gripper moving at high speeds, whereas approaching to place would have jaws closed and the gripper would be moving at high speeds. Also intuitively, each of these subtasks would require a different behavior of camera movement. For instance, the camera should be still while picking and placing the ring but should be continuously moving for the other to subtasks.

Moreover, it intuitively seems that these subtasks would have different movements of the camera with respect to the gripper motion. This is one of the investigations of this thesis where the behavior of the camera movement would be identified for each of these subtasks and compared.

5.3.1 Labeling of classes

While a participant was performing the exercise, I was annotating the subtasks during the task. Same video feed was displayed on the computer screen and using keyboard bindings, it was possible to annotate the changes in the subtasks. Only visual cues were used to label the data. The labeling was done using the definition as below:

- **Unknown Subtask:** When the user is getting ready and hasn't yet started the exercise, the default label is set to unknown subtask. Its corresponding

label is zero. As soon as everything is ready from the user and they start to move the gripper in the direction of the ring, the subtask changes.

- **Approaching Pick Up:** As soon as the user starts moving in the direction of the ring, this subtask is activated until the user is just about to close the jaws of the gripper to grab the ring. In this subtask the gripper motion is mostly at a constant height and with the jaw open.
- **Picking Up the ring:** When the user closes the jaws of the gripper to grab the ring and lift it to get it out of the peg or get it high enough to be able to move above the pegs. Once the vertical motion is completed, the user starts moving horizontally and the subtask changes. In this subtask, the motion of the gripper is mostly vertical with the jaws closed.
- **Approaching to place the ring:** After picking up the ring, the user starts moving in the horizontal direction towards the target peg. When the user is near the target peg they slow down and hover the gripper with ring just above the peg. In this subtask, the motion is mostly horizontal with the jaws closed.
- **Placing the ring:** As soon as the user is near the target peg, they try to align the ring on the peg and may swing the gripper to make the ring align on the peg and then release the ring so that it falls on the peg. In this subtask, the motion is mostly re-orienting the gripper to align ring on peg and then release the jaws. After doing so, it was observed that almost always the users will move away from the peg and ring to ensure the gripper is not in the way. Once, the ring is placed, a new target peg is selected and the process is repeated.

While labeling live when the user's performed the exercises, there was almost

DATA COLLECTION

always a delay in labeling due to which the labels were annotated wrong for the motion of the gripper. Hence post collection of data, the videos were analyzed and a code written using OpenCV to go through the recorded videos (at fps) frame by frame. The data was relabeled using the time stamps recorded at the bottom of each frame and the transitions between each subtasks were marked. Then these transitions were parsed again and data in between the two time stamps were relabeled accordingly. This relabeled subtask was stored as an additional column of the csv data file recorded during the user study for each of the file.

Chapter 6

Learning automated endoscope control

Chapter 3 introduced the methodology that this thesis uses to automate the endoscope. The reinforcement learning and inverse reinforcement learning algorithms were described in detail. However, how these algorithms are used and applied for automation of the endoscope was only briefly touched in that chapter.

This chapter shows in detail how Algorithm 2 is implemented. All the variables from the algorithm including the state space, actions of the autonomous and transition models for the Markov Decision Process are defined in view to automate the camera motion. A detailed discussion on the feature basis set chosen and the reasoning behind the choice is presented.

Furthermore, a metric to validate the obtained results from inverse reinforcement learning is defined. This is essential because it is difficult to quantify an automated trajectory to be good or bad since it is a subjective view. Therefore, it becomes necessary to introduce a metric quantifying the automated motion. This metric is based on similarity of the automated trajectories obtained with the human

trajectories collected from the user study.

6.1 Inverse Reinforcement Learning Model

From the previous chapters it can be seen that the simulation environment was designed such that the MTM can control a gripper and that gripper is used to pick up a ring and place it on a peg. The camera is moved by the user with their head movements mapping the movements of the simulated camera. These movements were recorded in the joint space of the endoscope and the cartesian space along with the movements of the gripper.

6.1.1 State Space

State space as defined in Chapter 3 is the set of all possible configurations of a representation of an environment that the agent reacts to. With this knowledge, we can define that the agent is the endoscope camera and the environment is everything that is seen through the camera. However, it is hypothesized that everything seen through the camera is not relevant to the motion of the camera but only the motion of the gripper is. The motion of the gripper encodes all the necessary information that is required for the camera movement. For instance, if the gripper is approaching to pick up the ring, the gripper will be moving in the direction of the ring and would be slowing down as it is near the ring. From this, it can be estimated where the ring is without knowing the position of the ring. Hence the environment can be compactly represented as the position and velocity of the gripper relative to the camera frame.

Fig. 6.1 shows the state of the environment in the image obtained from the endoscope camera. The state is the position of the gripper in this image where

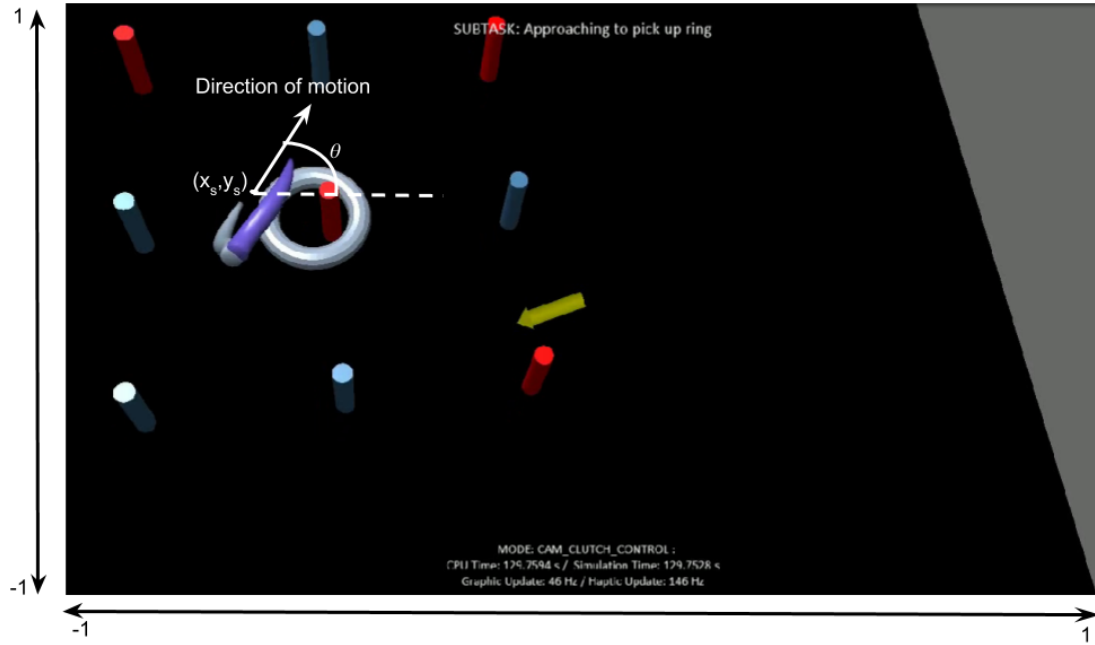


Figure 6.1: Figure showing the state of the gripper shown in the image of the camera view. The state of the environment is considered to be the gripper position in the image axis and the velocity of the gripper movement represented in the polar coordinates.

the axes of the image are scaled between -1 and 1. The state is represented as a 4 dimensional vector with two dimensions being the position and the other two are the velocity of the gripper in the image plane represented in polar coordinates. This is depicted by the arrow in the above figure.

To completely represent a dynamic model of the current simulation environment the current joint angles of the endoscope should also be included in the state space. This would contain the information of the camera pose and along with the gripper information, the state of the environment can be completely written. Also using this, the transitions for a particular action taken could be definitely calculated. However, it is not possible to include the information of joints of the ECM since the state space then becomes too large to be able to compute the value functions for a given reward iteratively. Since the reinforcement learning algorithm used is dynamic

programming algorithm, the computation time for it to perform n iterations grows exponentially with respect to the number of the states. Additionally, it would require a lot of memory to store a dense 8-dimensional state space. Considering these factors, the state space was considered to be only 4 dimensional. This state space allows to appropriately define the state of the environment and the rewards on them. However, the transitions become probabilistic rather than deterministic due to lack of information of camera pose. This is demonstrated in the following sections. This section will show the calculations of representing the state of the environment given the pose of the gripper in the simulation world frame.

Formally, let $\mathbf{p}_g^w, \mathbf{v}_g^w$ denote the 3D vector position and velocity of the gripper in the world frame and T_c^w represent the transformation matrix of the camera in the world frame (T_w^c represents is the inverse of T_c^w). Then the gripper position in the camera frame can be written as ¹,

$$\begin{aligned}\mathbf{p}_g^c &= T_w^c * \mathbf{p}_g^w \\ \mathbf{v}_g^c &= T_w^c * \mathbf{v}_g^w\end{aligned}\tag{6.1}$$

where $\mathbf{v}_g^w(t) = \frac{\mathbf{p}_g^w(t) - \mathbf{p}_g^w(t-1)}{\delta t}$ and $\mathbf{p}_g^w(t)$ is obtained from the simulation/data collected. However, this representation for the state space is 6 dimensional with no limits on any of the dimensions. Therefore it is difficult to discretize the space and apply Algorithm 1 with even the discretized space will be computationally very heavy in terms of memory usage and time to converge since it is a dynamic programming algorithm.

To reduce the dimensionality, the state was considered to be the projection of

¹Position vector is appended by 1 and velocity vector is appended by 0 to match the dimensions of the transformation matrix

the 3D world to a 2D image with respect to the camera. This is achieved using a projection matrix T_p given by,

$$T_p = \begin{bmatrix} \frac{1}{\tan(fov/2)} * \frac{h}{w} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan(fov/2)} & 0 & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \quad (6.2)$$

where fov is the angle of field of view, f and n are distances of far and near clipping planes respectively, h and w represent the height and width of the screen respectively. Now the gripper can be written in the image coordinates as,

$$\begin{aligned} \mathbf{p}_g^i &= T_p * \mathbf{p}_g^c \\ \mathbf{v}_g^i &= T_p * \mathbf{v}_g^c \end{aligned} \quad (6.3)$$

where \mathbf{p}_g^i and \mathbf{v}_g^i represent the coordinates of the gripper in the image. With the vector $\mathbf{p}_g^i = [x_g^i, y_g^i, z_g^i, w_g^i]^T$, the position of gripper on screen can be written as,

$$\begin{aligned} x_g^s &= x_g^i / w_g^i \\ y_g^s &= y_g^i / w_g^i \end{aligned} \quad (6.4)$$

And the velocity be derived using the chain rule for differentiation,

$$\begin{aligned} \dot{x}_g^s &= \frac{\dot{x}_g^i}{w_g^i} - \frac{x_g^i \dot{w}_g^i}{w_g^{i2}} \\ \dot{y}_g^s &= \frac{\dot{y}_g^i}{w_g^i} - \frac{y_g^i \dot{w}_g^i}{w_g^{i2}} \end{aligned} \quad (6.5)$$

Furthermore the velocity can be represented in polar coordinates as an angle and magnitude. This makes it easy to put bounds on the velocity vector and discretize the state space.

$$\begin{aligned}\theta_g^s &= \arctan2(\dot{y}_g^s, \dot{x}_g^s) \\ v_g^s &= ||(\dot{x}_g^s, \dot{y}_g^s)||_2\end{aligned}\tag{6.6}$$

The state of the environment is now defined to be the vector $\mathbf{s} = [x_g^s, y_g^s, \theta_g^s, v_g^s]^T$ and state space is $S = \{\mathbf{s} | x_g^s \in [-1, 1], y_g^s \in [-1, 1], \theta_g^s \in [-\pi, \pi], v_g^s \in [0, 0.1]\}$.

This state space is discretized by a grid space of length 0.01 in all directions except for θ where it the state is discretized into 100 equally spaced subsections each of length $\pi/50$. This discretization is done keeping in mind the memory consumption and are dense enough to not affect the performance of the algorithm. This makes the whole state space a matrix of dimensions 201x201x101x11. A coarser state space results in the state of environment not changing even when a significant translation of gripper has been done. A finer state space will linearly increase the memory usage and the computation time for the algorithm to converge. Different values for each dimensions separately were tried and tested and this value seemed to offer a good trade off between efficiency and computational time and hence was chosen.

6.1.2 Action Space

The agent based on the state of the environment performs an action. To automate the camera, these actions are defined to be the change in joint angles of the endoscope camera. Actions are considered to be discrete actions performed in discrete time. This is done to fit the current framework and algorithm for inverse reinforcement

learning. If a continuous action is to be performed, the algorithms would need to be tweaked to include integration in place of summation which is computationally expensive and not trivial to solve by hand. In simulation, the endoscope is controlled directly by the joint positions and the discretization and the time steps are small enough to make the movement of the camera feel continuous. Also for the hardware, the controller is a discrete controller operating in discrete time steps. Hence, this discretization does not impede the algorithm to be implemented on the hardware. Moreover the algorithm presented operates at a higher level sending commands at regular intervals to the lower level controller running at much higher speeds.

As described in earlier chapters, the endoscope has four degrees of freedom: yaw, pitch, insertion and roll. The actions for each time step for the yaw and pitch angles were selected as $\{-0.01^{\circ}, 0^{\circ}, 0.01^{\circ}\}$ step and the insertion as $\{-0.001mm, 0mm, 0.001mm\}$. The roll angle however was kept constant because during the user study, the users were asked to not roll their heads. If such an action was done the lens distortion effect of the headset would cause an instant motion sickness and headache. Therefore in the trajectories collected, the roll angle has hardly changed. Also, intuitively the this degree of freedom is not important for this particular task of pick and place. It is generally used to align the working axis for tasks such as suturing along the horizontal axis. Since this is already done here, this degree of freedom can be ignored.

Another reason to ignore the roll angle from the action set is due to the memory usage. If the roll is considered, that would result in 81 actions. Therefore, the corresponding policy, that specifies the probability of each action for each state, would be of 14.54 GB and would require this much amount of RAM to be loaded. Considering there would be 4 such policies for each of the subtasks, it would require special computers to run the algorithm leaving the usage of the proposed method

limited.

The action space now can be defined as $A = \{[\delta q_1, \delta q_2, \delta q_3]^T \mid \delta q_1 \in \{-0.01, 0, 0.01\}, \delta q_2 \in \{-0.01, 0, 0.01\}, \delta q_3 \in \{-0.01, 0, 0.01\}\}$. Therefore there are a total of 27 actions that can be taken at any time step.

6.1.3 Transition

The state as defined above is composed of a 2-D position vector and a 2-D velocity vector represented in polar coordinates. Using this, the new position of the gripper in the state can be defined as,

$$\begin{aligned} x_g^s &= x_{g,t}^s + v_g^s \cos(\theta_g^s) \\ y_g^s &= y_{g,t}^s + v_g^s \sin(\theta_g^s) \end{aligned} \tag{6.7}$$

This is not the next state since the action taken changes the joint angles of the endoscope which changes the matrix $T_{c,t}^w$ to $T_{c,t+1}^w$. Since, the state doesn't store the joint angles of the endoscope, these matrices are unknown and hence it is not possible to calculate the next state. If the joint angles and the $z_{g_t}^s$ were known, then the system would be a deterministic system, i.e. each state-action pair would result in a unique value for the next state.

From the trajectories collected it was observed that $z_{g_t}^s$ varied according to a Normal distribution with mean μ and variance σ as can be seen in Fig. 6.2. To calculate the probability density of the next state, the $z_{g_t}^s$ were sampled from this distribution and the current joint angles were uniformly sampled within their respective joint ranges. The next state was determined for each of these samples and a probability distribution was fitted on the resultant samples of next state.

Given the current joint angles (from samples), the action taken, the state (after

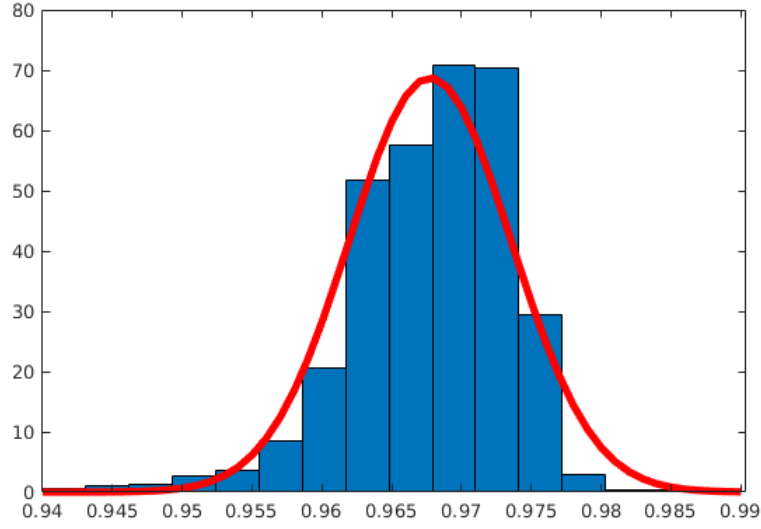


Figure 6.2: Histogram showing the variation of $z_{g_t}^s$ for the trajectories obtained during the data study along with the Normal Distribution curve (in red).

using Eq. 6.7) and $z_{g_t}^s$ (from samples) for a particular time t , the next state is given as,

$$\delta_i = [T_p T_{w,t+1}^c T_{c,t}^w T_p^{-1} - I] \begin{bmatrix} x_g^s \\ y_g^s \\ z_{g,t}^s \\ 1 \end{bmatrix} \quad (6.8)$$

$$\delta_s = \delta_i - \delta_i(4, 1) \begin{bmatrix} x_g^s \\ y_g^s \\ z_{g,t}^s \\ 1 \end{bmatrix} \quad (6.9)$$

$$\begin{aligned}x_{g,t+1}^s &= x_{g,t}^s + v_g^s \cos(\theta_g^s) + \delta_s(1, 1) \\y_{g,t+1}^s &= y_{g,t}^s + v_g^s \sin(\theta_g^s) + \delta_s(2, 1)\end{aligned}\tag{6.10}$$

6.2 Trajectories

Data collected in user study were transformed from the gripper position using camera joint angles recorded into state and action representation. Since the data was recorded at 1000 Hz, there was an abundance of data and would slow the algorithm down. To solve this, the trajectories generated from the state-action representation were generated at 100Hz. This particular frequency was chosen keeping in mind that the frequency that the headset transmits data is only 50 Hz. However if the data was down-sampled to 50 Hz, the resultant gripper transitions would be too big for a time step. Also, the data set would have been too small for training. Therefore the next multiple of 50 Hz was chosen.

Trajectories were segmented based on the classes of subtask they were labeled in. Therefore for different subtasks different “expert” trajectories were recorded and inverse reinforcement learning performed to determine policies corresponding to each task. The number of trajectories however were different for different classes since there were some subtasks which were performed more number of times. During some of the trials, the users would drop the ring before the ring was placed on the target peg resultant in the number of trajectories generated for approaching picking and picking to be more than approaching place and placing. The number of trajectories for each of the subtasks are shown in Table 6.1. Also, the trajectories for each subtask all differ in length. It was observed from the user study that the users spend the most time in placing the ring over the peg. This is understandable

since the ring would be vertical while users carry it to the peg and would want it to be horizontal to place it. To make it horizontal they have to move the gripper in a to and fro motion so that the ring would become horizontal due to the inertia and could then be placed. This motion is undesired though and as discussed in the next chapter affects the performance of classification of subtask.

Table 6.1: Table showing the number of trajectories for each of the subtasks.

| Subtask | Number of trajectories |
|----------------------|------------------------|
| Approaching pick up | 274 |
| Picking Up | 274 |
| Approaching to place | 246 |
| Placing | 246 |

6.3 Feature Set

The reward function to be determined is defined as linear combination of a basis feature set of functions ϕ_i . Determining the weights of these feature basis set is the goal of the inverse reinforcement learning. Hence, defining good features is important to the performance of the algorithm. In total there were 12 features, all Gaussian functions with a variance of 0.3 were chosen. These features could be broadly classified into two categories: one where mean of the Gaussian is constant, and one where the mean changes with the state. The thinking behind choosing these feature vector is to try and cover the whole state space but still have a small amount of features considering the memory required. Therefore a relatively large variance was chosen.

For a state $(s) = [x, y, \theta, v]^T$, the first class of feature functions can be represented as,

$$\phi_1(s) = \exp\left(-\frac{(x - \mu_x)^2 + (y - \mu_y)^2}{0.03}\right) \quad (6.11)$$

where $\mu_x = \{-0.5, 0, 0.5\}$ and $\mu_y = \{-0.5, 0, 0.5\}$. A combination of these means result in a total of 9 Gaussian functions that are used as the feature vectors. To better understand this class of feature vector, consider a Gaussian centered around the center of the image ($\mu_x = 0$ and $\mu_y = 0$). If the weight for this feature function is high, this would result in policy being such that the camera viewpoint would have the gripper always in the center of the frame eventually.

The second class of feature vectors used have a mean that is depending on the state, in particular the velocity of the gripper and can be represented as,

$$\phi_2(s) = \exp\left(-\frac{((x + nv \cos(\theta))^2 + (y + nv \sin(\theta))^2)}{0.03}\right) \quad (6.12)$$

where $n = 1, 2, 3$. An intuitive reasoning for the second class of features is that if the gripper is moving, the center of the frame would be where the gripper would be in the future time steps. That is, it is trying to look ahead of the motion of gripper suggesting that the user might be looking at an object which the gripper is trying to reach (for eg. the ring or the peg).

6.4 Policy

A policy which defines the probability of taking an action given a particular state. Once this policy has been obtained using Algorithm 2, policy can directly be used to control the endoscope motion. The gripper motion can be converted to a state value using the calculations given in the above subsections and an action which has the highest probability would be performed. This can be done at each time

step to give a continuous motion of the endoscope and generate a trajectory. This trajectory of the endoscope motion can be represented in joint space, $Q = \{\mathbf{q} = [q_1, q_2, q_3, q_4]^T \mid q_1 \in [-\pi/2, \pi/2], q_2 \in [-\pi/2, \pi/2], q_3 \in [0, 0.2], q_4 \in [-\pi, \pi]\}$.

6.5 Initial State Visitation Frequency

In Algorithm 2, the state visitation frequency needs to be calculated for the optimal policy obtained and is calculated iteratively. An initial estimate of this state visitation for each state has to be provided which is updated using the optimal policy obtained. This estimate is provided using the collected expert trajectories. All states visited in the trajectories were plotted and the histograms with the fitted distributions are shown in the Fig. 6.3.

The distributions for each of the variables in the state is summarized in the Table 6.2. For the variable θ of the state as it can be seen from Fig. 6.3, the probability distribution is a mixture of two Gaussian functions with equal weights.

The probability of visiting any state $(s) = [x, y, \theta, v]^T$ given this information can now be obtained by multiplying these individual probabilities, i.e.,

$$\mu_0(s) = \frac{P(x)P(y)P(\theta)P(v)}{Z} \quad (6.13)$$

where Z is the normalizing constant.

Table 6.2: Table showing the initial state value frequency distribution for each variable of the state space.

| Variable | Distribution | Parameters |
|----------|------------------|---|
| x | Normal | $\mu = 0.15, \sigma = 0.25$ |
| y | Normal | $\mu = -0.27, \sigma = 0.25$ |
| θ | Gaussian Mixture | $\mu_1 = -\pi/2, \mu_2 = \pi/2, \sigma_1 = 0.6, \sigma_2 = 0.6$ |
| v | Exponential | $\lambda = 0.004$ |

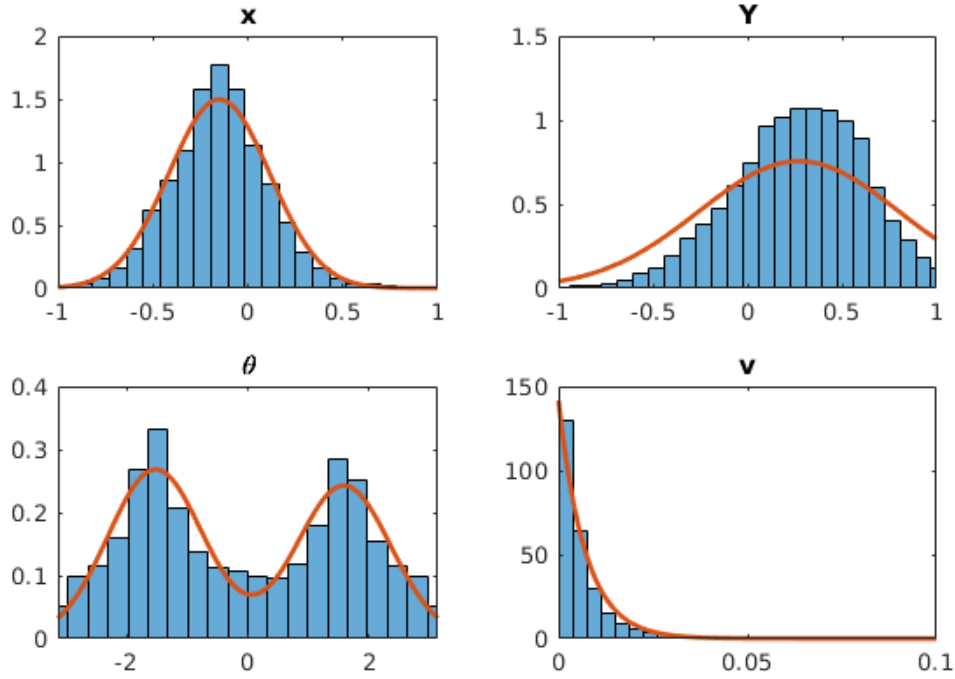


Figure 6.3: Histogram showing the frequency with which states were visited in the collected data trajectories. Four histograms are plotted for each variable of the state along with the probability distributions (in red).

6.6 Validation of learned rewards

In this thesis, the problem statement is to automate the endoscopic camera to react to the gripper motion in a manner which is similar to how a human would move the camera. The assumption here is that the human inherently knows the rewards/penalties of the camera movement and moves the camera optimally according to that. The inverse reinforcement learning algorithm presented in the earlier section gives an estimate of the reward function and a policy. Since the actual reward function remains an unknown, there is still a need of a metric to determine how well the algorithm works.

The estimated reward function can be evaluated based on the corresponding policy obtained. A trajectory then can be generated from this policy by taking

actions dependent on the state which is the gripper motion. Once an automated trajectory is generated for a test data set of gripper for which human generated trajectory is also available, the two trajectories can be compared.

For comparison between trajectories, a similarity metric is defined as, $\sigma : Q \times Q \rightarrow [0, 1]$ between the human operated camera trajectory $\tau_H = [\mathbf{q}_1^H, \mathbf{q}_2^H, \mathbf{q}_3^H, \dots]$ and the automated trajectory $\tau_A = [\mathbf{q}_1^A, \mathbf{q}_2^A, \mathbf{q}_3^A, \dots]$ is defined as,

$$\sigma(\tau_A, \tau_H) = e^{-\sqrt{\frac{1}{T} \sum_{t=1}^T \|\mathbf{q}_t^H - \mathbf{q}_t^A\|_2^2}} \quad (6.14)$$

where $\|\cdot\|_2$ denotes the norm of the vector \mathbf{q} in the joint space. Since, this is an exponential function the output value would be between 0 and 1, with 1 being exactly the same trajectory and 0 being the similarity of two infinitely different trajectory. With this metric defined, the quality of a policy obtained from the inverse reinforcement learning can be quantified with respect to a human generated trajectory.

Chapter 7

Results and Discussions

This chapter shows the results obtained from various approaches presented earlier in the thesis and discusses them thoroughly regarding their impact towards the objective of this thesis.

7.1 Subtask classification

The classification of the task into 4 subtasks using the time series motion data of the gripper data was done using the long short term memory recurrent neural network as described in Chapter 3. The LSTM network was chosen due to its ability to selectively remember the inputs and predicted outputs of the system and use those to generate predictions for the current time. The data obtained from the user was labeled into the 4 subtasks as described in Chapter 5 by going through the videos recorded frame by frame and using the simulation time in the frame to decide the transitions down for classification. The rules to classify a frame into the subtasks are also specified in Chapter 5.

In order to maximize the accuracy obtained from the neural, parameters such as inputs, number of hidden layers, number of neurons in each hidden layer, activation

RESULTS AND DISCUSSIONS

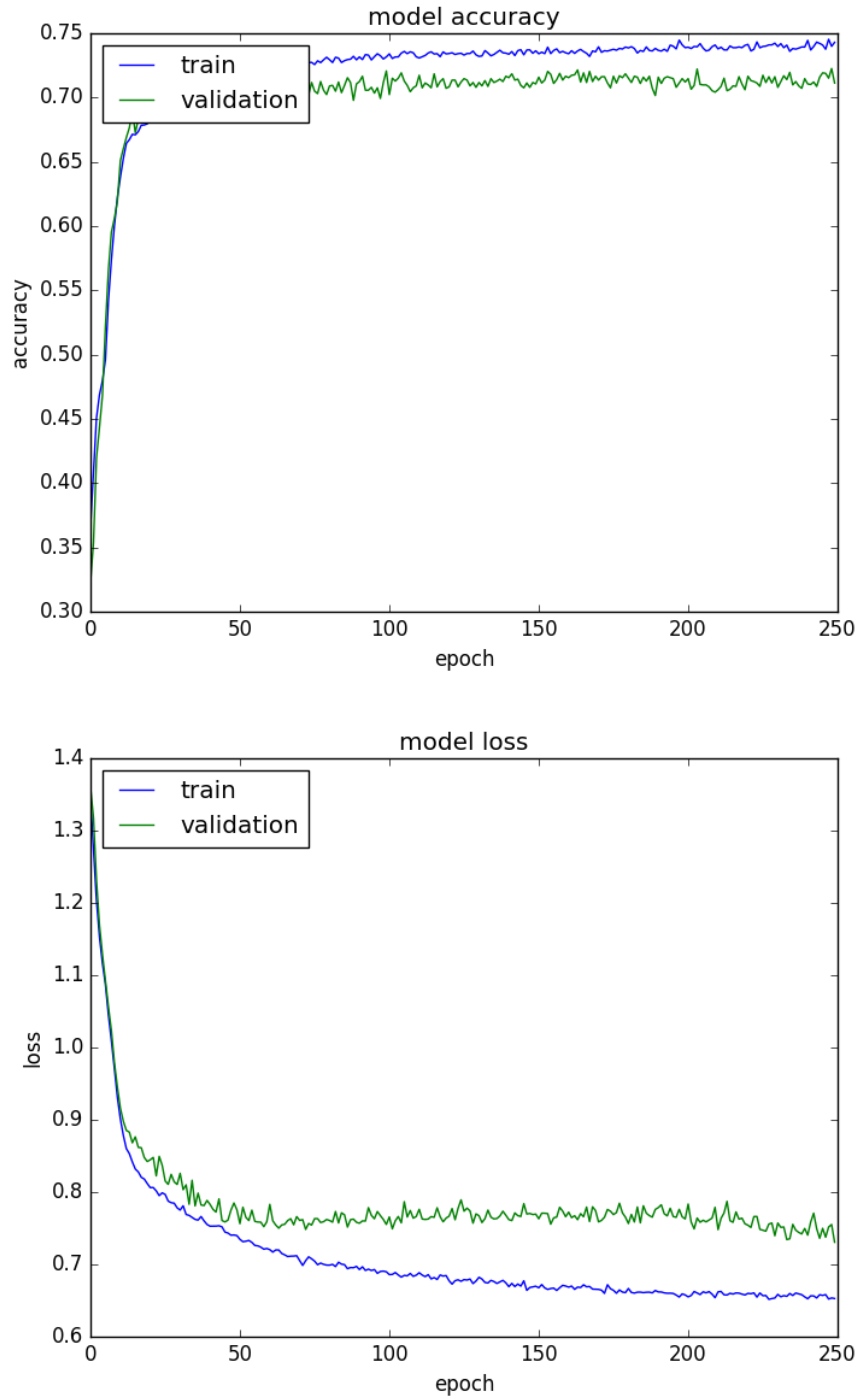


Figure 7.1: Graphs showing the accuracy and loss for the implemented neural network evolve over the number of epochs for train and validation data sets. A training accuracy of 74.5% and a validation accuracy of 71.3% was obtained.

RESULTS AND DISCUSSIONS

functions and the optimization function (stochastic gradient descent, adam) should be varied and compared. A lot of different trials were required for the combination of each of these parameters. These trials were performed and the best results were obtained with the following parameters. All of these trials were implemented using the keras library [79] using Tensorflow [80] the backend for computation.

Inputs to the neural network were given as 4 dimensional vector including the position of the gripper and its jaw angle. This 4-d vector was concatenated with 10 previous time stamp values to contain the information of the derivatives of the inputs. Hence, the total inputs for a sample given to the neural network was a 10×4 matrix. Inputs with differing number of previous data to be used were also tried along with using the orientation of the gripper to classify. However, these resulted with lesser accuracy or overfitting, where the training accuracy was very high but the validation accuracy was low showing the network did not generalize well.

The optimizer used was the Adam optimizer [77]. The parameters resulting in best performance were a learning rate of 0.0007, $\beta_1 = 0.0999$ and $\beta_2 = 0.0999$. Again, these parameters were fine tuned by running the algorithm for multiple trials with other values of these parameters. Other optimizer that was tried was the stochastic gradient descent [78] but the Adam optimizer resulted in better accuracy and converged faster.

The best results for validation accuracy is shown in Fig. 7.1. This figure shows the evolution of the training and validation accuracy along with the loss in error with respect to the epochs for the training of the classification. It can be observed that the training accuracy is 74.5% and the validation accuracy is 71.3%. This shows that the trained model does overfit or underfit the data and hence the model represents a good generalization of the data.

The structure of the neural network used had a single hidden LSTM layer con-

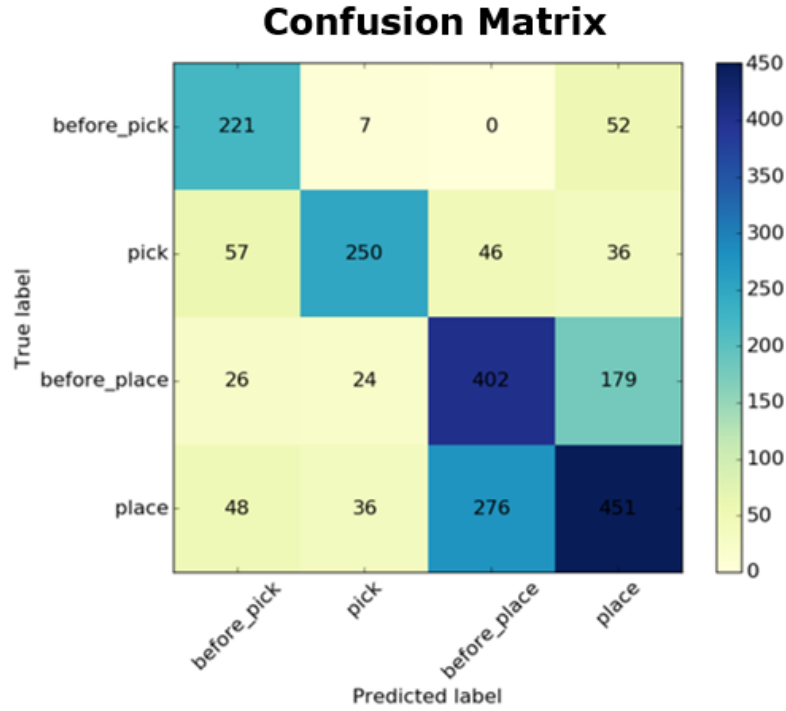


Figure 7.2: Confusion matrix for the subtask classification generated for a completely unseen data set. It can be seen that the accuracy is reduced due to confusion between subtasks 3 and 4.

taining 10 LSTM cells in addition to the input and the output layers. The input layer had a linear activation function so that the LSTM layer would get the inputs as they were without alterations. The output layer had a softmax activation function for classification into the 4 subtasks.

For a better understanding of the accuracy and generalization of the trained model, Fig. 7.2 shows the confusion matrix for a completely unseen test data set. This data set was not used by the neural network for either training or validation. The confusion matrix depicts the comparison between the true labels and the predicted labels. It can be observed from the figure that the model classifies the first two subtasks with a very good accuracy. However, for the subtask 3 and 4, the algorithm is not that accurate and is confused.

A possible explanation for this is that when the user is about to place the ring on the peg, the ring needs to be approximately horizontal above the peg. However, when the gripper is carrying the ring, it is vertical in orientation due to there being only one gripper to hold the ring. Therefore to align the ring, the users have to move the gripper horizontally in a to and fro motion at larger speeds so that the inertia of the ring makes it horizontal. This motion data given to the neural would make it difficult for it to determine as to whether the user is approaching to place the ring or placing the ring. To resolve this issue and increase the accuracy, either a relabeling of the data is necessary or filtering out this part from the gripper motion should be done. More discussion in ways to improve the accuracy of the classification is done in the next chapter.

7.2 Automation of camera motion

The inverse reinforcement learning (IRL) algorithm described in Chapter 3 was run using the parameters and variables as described in Chapter 6. This section provides the results of the algorithms showing the reward and value functions obtained.

7.2.1 Convergence of inverse reinforcement learning

The IRL algorithm is an optimization algorithm using gradient ascent to maximize the likelihood of a number of given expert trajectories. The probability or the likelihood of a trajectory has been assumed to be directly proportional to the exponential of the rewards collected during the trajectories. The objective is to find the rewards that maximizes the likelihood of these trajectories. For this, a set of basis feature vectors were defined and a weighted sum of these vectors were considered to be the reward function. Based on this, the gradient of the objective was calculated and the

weights were altered

It should be noted here that these weight vectors are unit vectors that represent how each of the features would be weighed with respect to each other. If the constraint of a unit vector is not enforced, then the objective cannot attain a maximum for the rewards as the rewards for the trajectories will get larger in proportion to the weights in the optimal direction.

This algorithm was run for 10 iterations and it was observed that it converged very fast to the optimum set of weights for all classes. Since all possible trajectories for the given MDP cannot be evaluated, therefore the probability of the expert trajectories cannot be explicitly calculated and depicted to show the convergence of the algorithm. However, the rewards obtained by these expert trajectories during each iteration can be calculated and is a good measure to show the convergence of the algorithm.

Fig. 7.3 shows the rewards obtained for the given trajectories evolving over iterations for each of the subtask classes (leftmost is class 1 and right most is class 4). It can be seen from these figures that the rewards for the trajectories reached a maximum in 1-2 iterations. The initial weights were randomized and had no bearings on the final weights. Moreover, over a number of trials with these initial random weights, the final optimum value of the weights obtained were the same showing the reliability of the optimization.

7.2.2 Reward Function

A reward function as defined in this thesis is a function of state of the environment based on which the agent reacts. The function gives the rewards/penalties of being in a state and hence defines the behavior of the agent. Here, the state space is a 4-dimensional space consisting of the gripper position in the image/screen coordinates

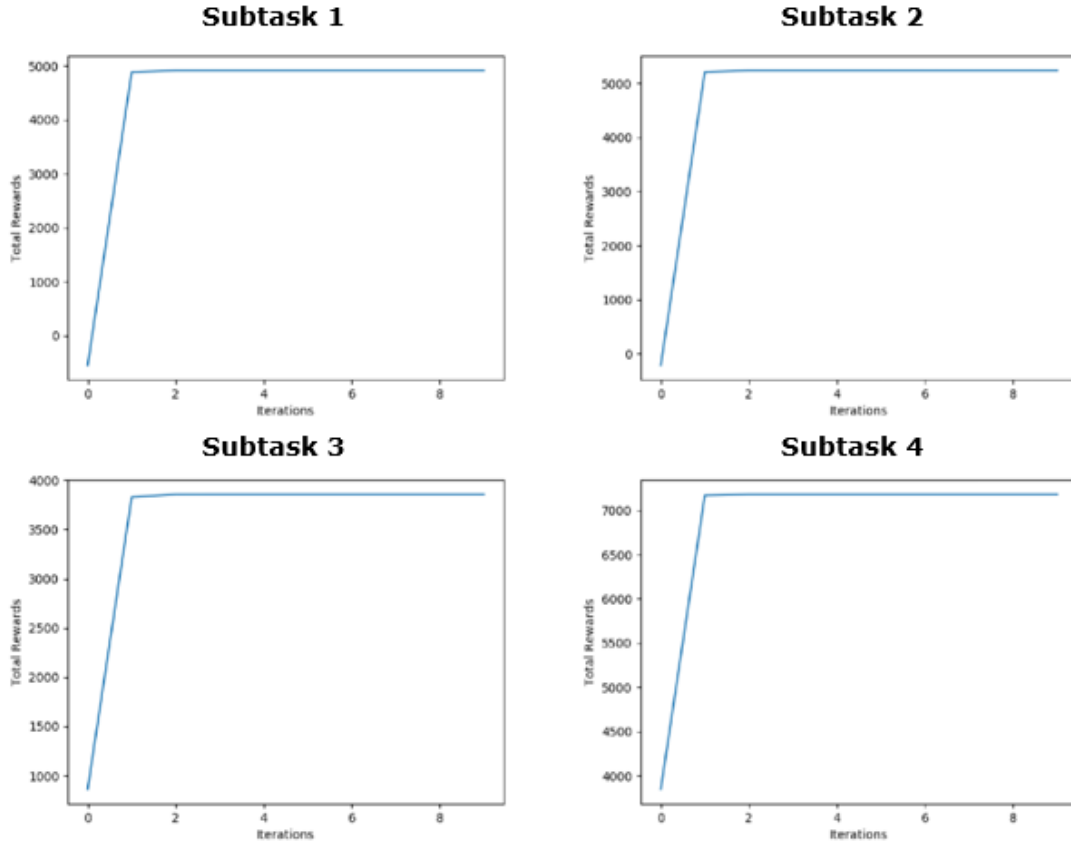


Figure 7.3: Rewards obtained by the expert trajectories evolving with the number of iterations of the IRL for all 4 subtask classes in order from left to right.

and the gripper velocity represented in polar coordinates in the image plane. The IRL algorithms calculates the optimal reward function as a weighted sum of 12 feature Gaussian functions with differing means based on the state.

For each of the iterations of the IRL algorithm, the computed reward functions are shown in the Fig. 7.4 for all classes. Each row in the figure depicts the evolution of reward function for each of the 4 classes from top to bottom respectively. The graphs are contour plots of the reward functions with darker colors representing lower values. The axes of the graphs represent the image/screen axes. The horizontal axes represents the x-direction and the vertical axes the y-direction. Since the state space is 4 dimensional, the graphs plotted here represent the reward function for the states

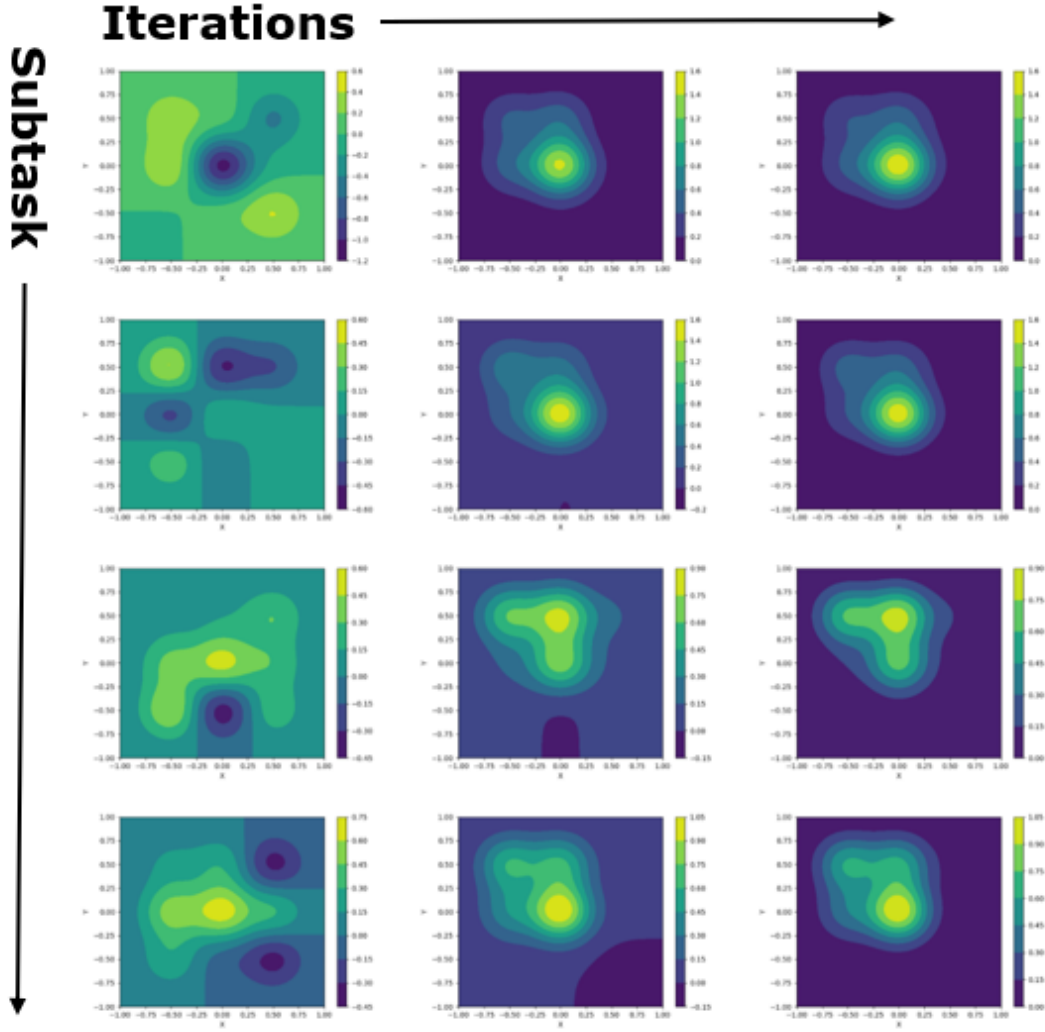


Figure 7.4: Reward function computed by the inverse reinforcement learning algorithm for the subtask 1. This figure shows the reward function for iteration 0 (left), 1 (center), 10 (right). Each row shows the reward function for the 4 classes of subtasks.

RESULTS AND DISCUSSIONS

where the gripper is not moving. Discussion regarding the gripper motion is done in the next subsection.

The first column of the figure shows the random reward functions resulting from the initial random weights used. The second column shows the how much the reward function has changed in one iteration and is almost identical to the optimal reward function shown in the third column. Subtle changes in the shades of contours can be seen from column 2 to column 3.

It can be observed from the first row that even though the initial weights for the Gaussians centered at the center were negative, the algorithm was efficient to correct them and show an expected maxima at the center of the screen.

For the subtask 1 which was to approach to pick up the ring, the reward function has a global maxima at the center of the image. This implies that whenever the gripper is still and the user is approaching to pick up the ring, the maximum reward would be achieved if the endoscope was moved such that the gripper is in the center of the screen. A similar observation can be made for the subtask 2 and 4.

However for subtask 3, which is to approach placing the ring on the peg, the reward function obtained only has a local maxima at the center and a global maxima which is above the center. This is probably due to the fact that that near the end of subtask 3, when the gripper is stable, the gripper is not in the center of the scene but is just above it as the target peg is in the center of the screen. A similar case can be made when the subtask is started, the gripper is just above the current peg, the ring has just been taken out and the gripper starts moving horizontally. Since the gripper is above the level of pegs during this whole subtask, the reward function is probably not at the center of the image but at above the center of the image.

Another observation that can be made from the contours are that these are slightly extended contours towards the top-left of the image plane. This behavior is

harder to explain although the contours are very dark indicating that the peaks are very small. These small peaks don't affect the computation of the value functions as shown in the next subsection.

7.2.3 Value Function

Value function as defined earlier in the thesis is the function that quantifies how good or bad a particular given state is. It is in some sorts an equivalent of a potential function in control theory. Though in this case we control the camera to go to a state with higher value at all times rather.

Value function is a superposition of the reward function and the dynamics of the system over time. The policy which is responsible for taking actions does so by taking an action so that the next state has the highest value amongst the possible next states for the current state. Hence to evaluate the behavior of the obtained policy, it suffices to observe and infer from the value function.

Fig. 7.5 shows the value functions calculated for the corresponding reward functions shown in Fig. 7.4. From the figure it is evident that highest value obtained when the gripper is stationary is when it is at the center of the image obtained from the endoscope (in the final value function). In other words, the algorithm would move the endoscope in such a way that the gripper is at the center of the image if not already. This is the case for subtasks 1, 2, and 4.

For subtask 3 however, the maximum value is seen at $(0,0.5)$ which is above the center of the image. This is expected behavior given the corresponding reward function. Hence, if the gripper is stationary during this subtask, the camera would move in such a way that the gripper would be directly above the center of the image.

Another observation that can be made from the figure is with respect to the radius of the yellow contour corresponding to the maxima of the function. For

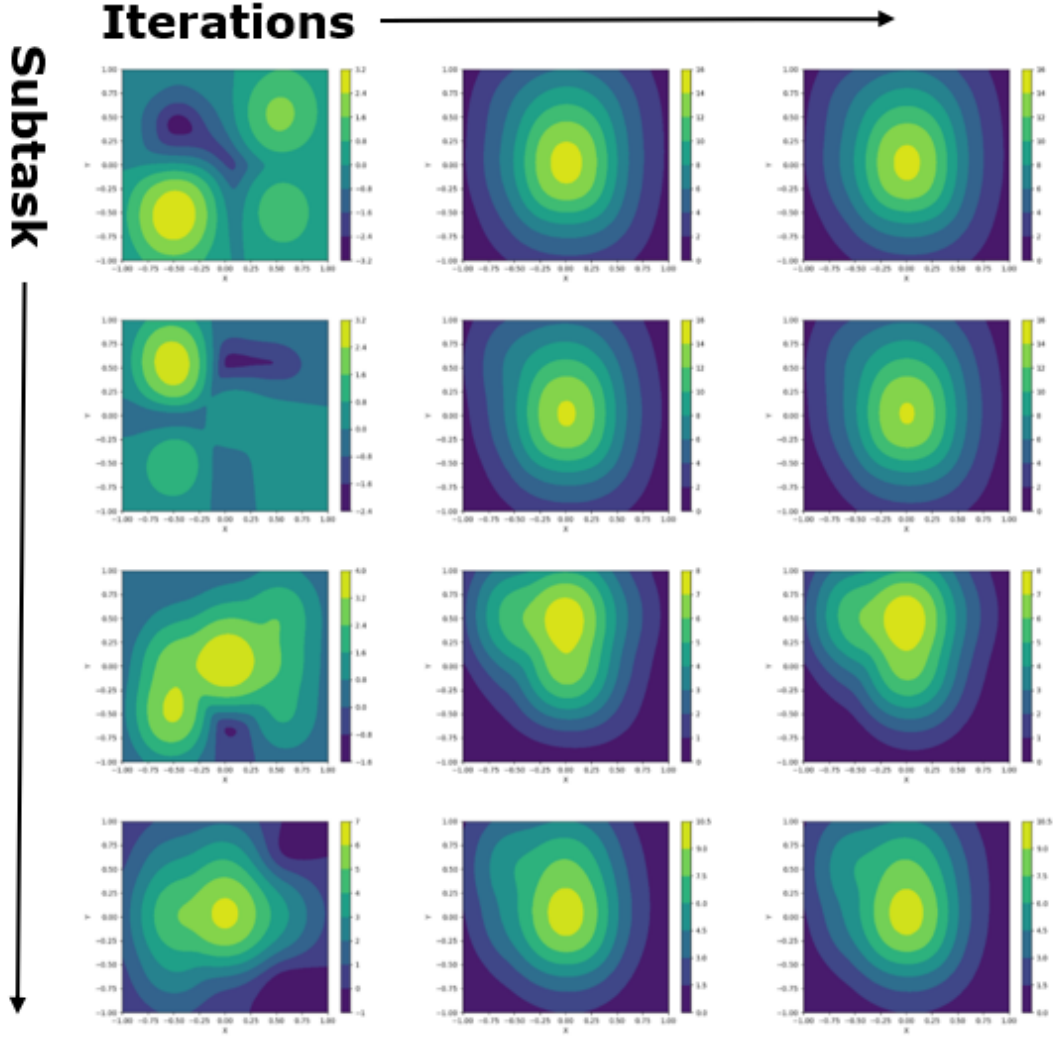


Figure 7.5: Value function computed by the inverse reinforcement learning algorithm for all subtasks (arrange by the row). Columns show the value function for iteration 0, iteration 1 and iteration 10 of the algorithm

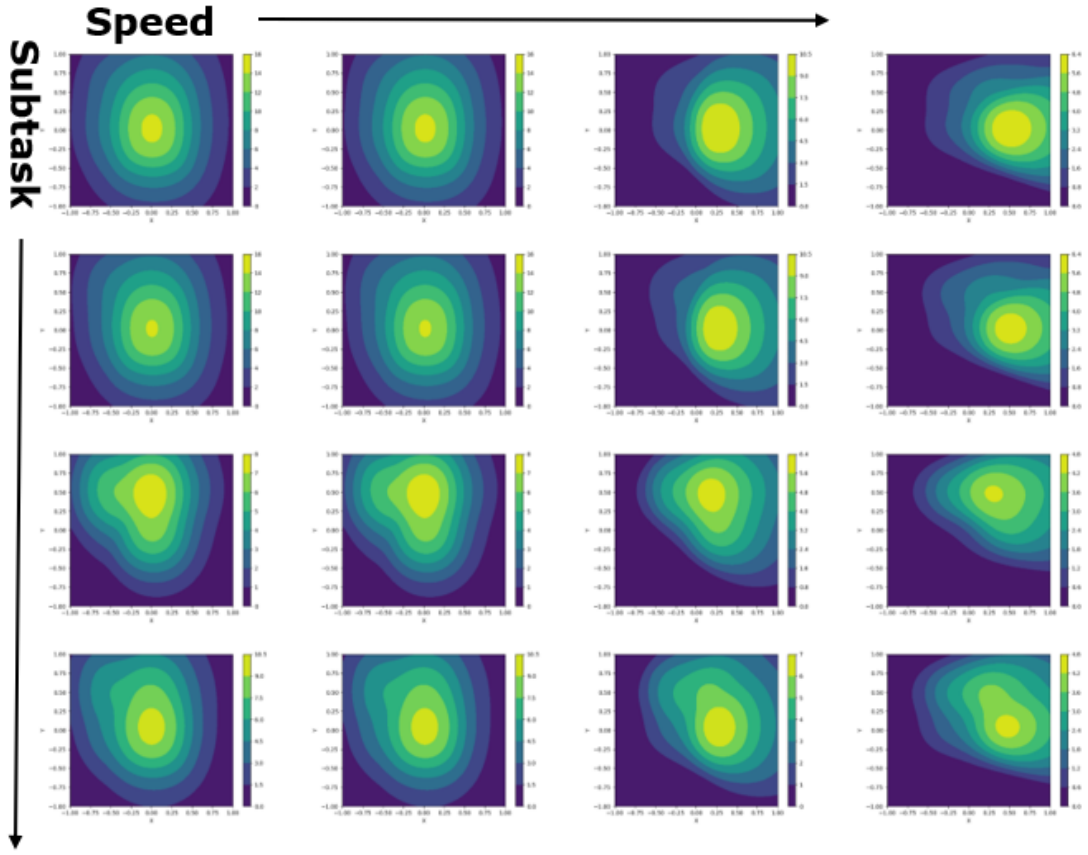


Figure 7.6: Final value function obtained from IRL plotted for different speeds of gripper motion in the negative x direction: 0, 0.01, 0.05, 0.1 units in order in columns for all subtasks (rows).

subtask 1, it is wider than compared to subtask 2. This implies that the camera would allow a larger movement of gripper without moving than in case for subtask 2 and hence would be more stable in this particular case for subtask 1. It would be even more stable for subtask 3 and subtask 4.

Discussing the movement of gripper, it is important to analyze the value function when the gripper would be moving as well to get a sense of how the algorithm performs. Fig. 7.6 shows the value function for varying gripper speeds when the gripper is moving in the negative x direction. From this figure, it can be observed that the maxima shift in the x-direction which is the direction opposite to the

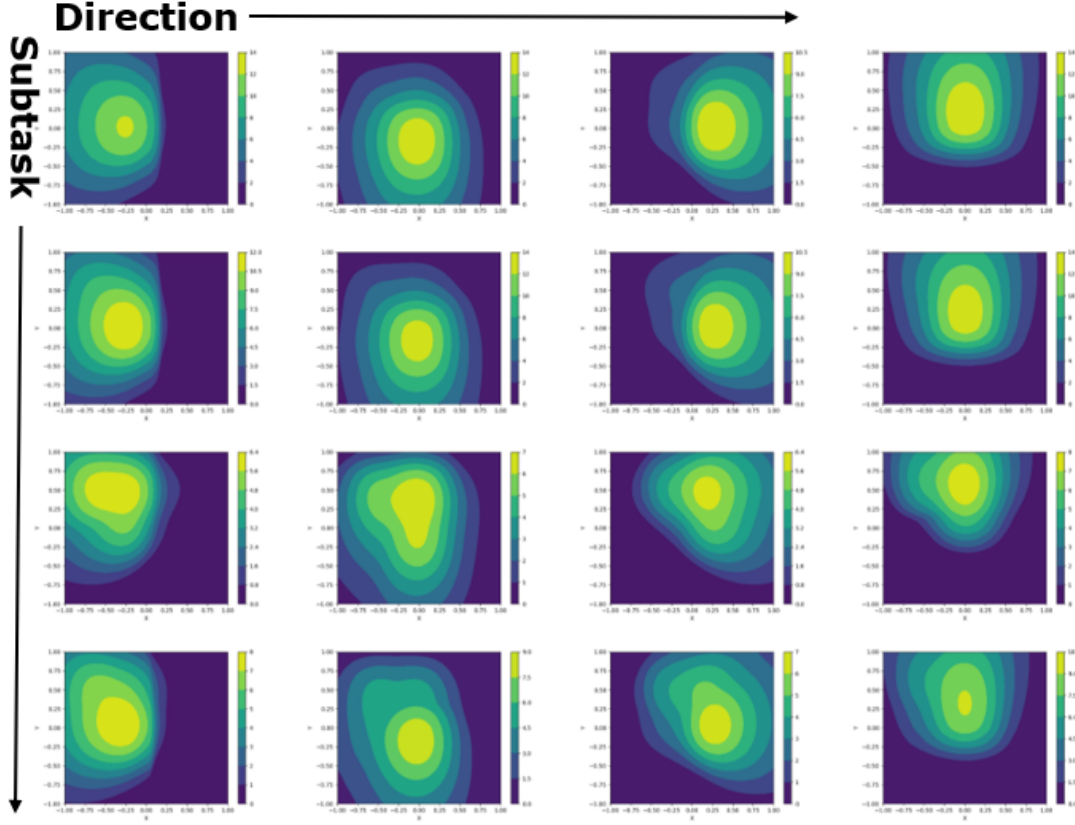


Figure 7.7: Final value function obtained from IRL plotted for different directions of motion of gripper at speed of 0.05 units: right, up, left, down in order in columns for all subtasks (rows).

movement of the gripper. The center of the image is ahead of the gripper in the direction of its motion and hence the algorithm moves endoscope to look ahead of when the gripper is moving. Even in the case of subtask 3, it is shifted from where it was when the gripper is stationary. This suggests a continuous motion of the camera for a continuous motion of gripper.

This figure shows the variation of value function for differing speeds of 0, 0.01, 0.05 and 0.1 from left to right in each row. It can be noted that as the speed increases, the maxima shifts more implying that the endoscope is looking more and more ahead of the gripper. It is almost like the endoscope predicts where the gripper

would be stopping and looks ahead so that there would not be sudden disruptive movements of the camera when the gripper moves. That is if the gripper is moving and stops, in all probability the camera would already be in position such that the gripper would be in the center of the image as suggested by the earlier figure.

This behavior is not only observed when the gripper is moving left but also in any other direction as can be seen from Fig. 7.7. The maxima always shifts in the direction opposite to the motion of the gripper so that endoscope can provide a view to look ahead to where the gripper is moving.

Hence the algorithms used to automate the endoscopic camera motion not only track the instruments but is also intelligent in tracking of these instruments. It is aware of the subtask the user is performing, the direction and speed of motion of the gripper and changes its behavior accordingly. It is probably very similar to how we move our heads when picking and placing objects which was the objective of this thesis. The comparisons between the automated camera trajectory and human camera trajectory obtained from head tracking in the user study is shown in the next section.

7.3 Similarity of trajectories

One of the main objectives of this thesis is to show that the approach presented results in an endoscope trajectory that is similar to the way an assisting surgeon or staff would have operated the endoscope. This section discusses the results and of similarity between trajectories collected during the user study and the ones generated by the algorithms presented earlier. These trajectories were not used to train the inverse reinforcement learning models or the neural networks for classification. They are completely unseen data to the algorithms presented and therefore are a

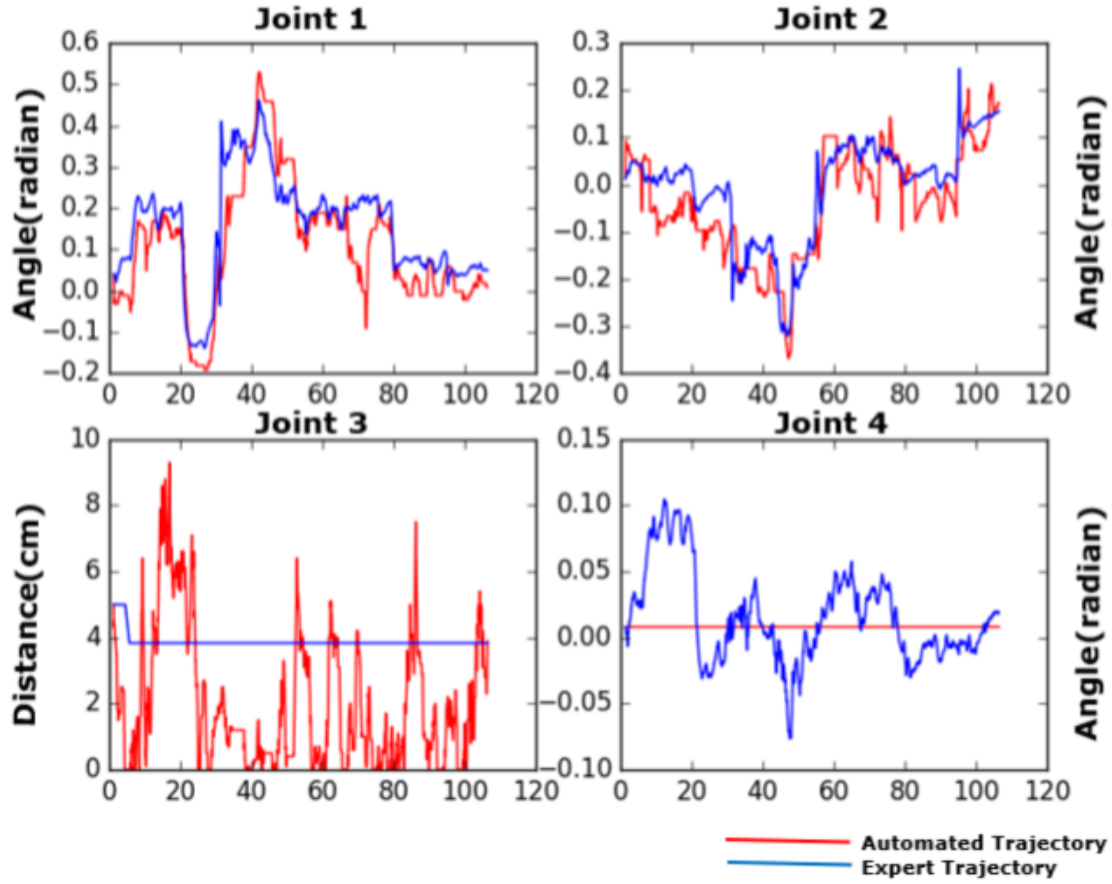


Figure 7.8: Comparison between the recorded human trajectories with the automated trajectory in the joint space of the endoscope for the same gripper motion data.

good representative of using the algorithm in practical scenarios.

The comparison between the two trajectories for the endoscope in joint space has been shown in Fig. 7.8. The red line shows the automated trajectory and the blue line depicts the trajectory collected for a single trial. It can be observed from the graphs that the automated camera trajectory follows the human trajectory for the yaw and pitch angles. Though for joint 3, which is the insertion degree of freedom and therefore is responsible for zooming and out the graphs are a very different.

It can be noted that the user hardly moves the insertion degree of freedom. This is due to the fact that this degree of freedom is not controlled by the virtual reality

RESULTS AND DISCUSSIONS

headset but by a foot pedal on the daVinci master console and therefore when the users are concentrating on completing the exercise, they do not bother to change the insertion using the foot pedal. It is another thing to remember for them and therefore adds to the cognitive load on the users. Thus we see minimal use of the foot pedal from the users.

In contrast, the automated trajectory shows a lot of movement in that joint. Since the inverse reinforcement learning tries to move the endoscope in order to maximize the reward, which is highest if the gripper is kept near the center of the screen (if it is not moving) it zooms out of the screen so that the movement of the gripper in the image is lesser for a given movement of gripper in the world frame. This helps stabilize the view.

From Fig. 7.8, it can be seen that the automated trajectory has no change in joint 4 because it was assumed that the user's don't roll their heads by a large amount. This assumption is certainly validated from the graph as it can be seen that the roll angle is always between -0.1 and 0.1 that is between -5 to 5 degrees.

Even though the joint 3 trajectories are not very similar, the similarity metric defined in Chapter 6 (Eq. 6.14) results in an average similarity of 94.68% on the 9 (out of total 45) trials that were not used for training or validation. This suggests that apart from the differences in joint 3, the automated trajectories computed are very similar to the human generated trajectories recorded during the user study.

To have an even better correlation between the human and automated trajectories, there are two things that could have been improved upon. If the data could have been collected by tracking the position of the virtual reality headset or by tracking the eye gaze by the headset itself (newer VR headsets do this), then there would have been no need of a foot switch to zoom in and out. This would have helped in the data being more representative of how and where users see while performing

RESULTS AND DISCUSSIONS

the task in terms of zooming in out. Also, while selecting features, there should have been features to include zoom in and therefore affect the reward functions calculated.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

The work presented in this thesis provides an approach to having an intelligent camera control for a pick and place task. As discussed earlier, the pick and place task is one of the fundamentals tasks surgeons use to practice their laparoscopic skills and is one of the basic tasks that require an extensive use of camera motion to constantly adjust viewing points.

The goal of this work was to provide an endoscopic camera control algorithm that is aware of the surgeon's intent and adapts to it rather than following a pre-defined set of rules. This goal was achieved by performing a user study to obtain a set of human trajectories showing the optimal viewpoints for the various stages of the task. These trajectories were given as input to train the inverse reinforcement learning algorithm to obtain optimal behaviors of the camera for each of the subtask of the task at hand. Moreover, the trajectories obtained from the algorithm were compared to the human trajectories of camera from the user study for a subset of data set that was not used for training the algorithm. A similarity of 94.68% was

CONCLUSION AND FUTURE WORK

obtained using the similarity metric defined based on the root mean squared errors of the trajectories.

The pick and place task was classified into 4 subtasks based on the motion data of the gripper. These subtasks represented the intent of the surgeon and were categorised keeping in mind the differences in camera motion during each subtask. The data obtained from the user study was then used to train a recurrent neural network using LSTM for classification of gripper motion data into subtasks. An accuracy of 71.3% was obtained on the test data set for the neural networks trained. On further inspection of the results of classification using confusion matrix it was observed that the algorithm was confused between two of the subtasks and a better labelling strategy could help resolve this issue.

In all, an approach for automating the endoscope camera motion for teleoperated robotic surgeries was shown in this thesis. This approach results in an intelligent camera control that is able to predict surgeon's intent and react to it. The algorithms presented were modelled on human data and extract information of how human operators handle the endoscopic camera motions. This allowed the camera to not only track the gripper motion but also was able to look ahead and show the objects of importance in the environment. This is an important result since the attention of surgeons is not always on the instruments but also on the surrounding objects and the algorithm is able to predict the positions of these objects by using just the kinematic data of the user.

Since automating endoscopic control for a generic task and situation has a wide scope due to the unique challenges of different procedures, the thesis only provides a way of automation for a fixed task. This leaves a lot of scope for works to build upon this thesis as discussed in the next section.

8.2 Discussion and Future Work

8.2.1 Improving accuracy of classification

A Long Short Term Memory cell was used with the recurrent neural network to classify the task of pick and place into 4 subtasks. This neural network however resulted in only a 71% accuracy in this classification. This accuracy can however be increased if the labels were defined more clearly and consistently. The neural network gets confused between whether the user is approaching the peg to place the object or is placing the ring on the peg. This is partly due to the dynamics of the rings from CHAI libraries due to which the users have to move the gripper at great speeds to get the ring to become horizontal so that it can be placed. However, generally while placing the rings, gripper does not have horizontal velocities but only vertical velocities. Therefore this confuses the network to classify at the transition of the two subtasks. For future tasks therefore, it is important that the classification is defined very clearly and precisely and labeling is done very carefully in order to avoid the confusion. Maybe, filtering the data a bit to reduce jerks can be done and would be helpful in reducing the confusion.

Moreover, another approach like Hidden Markov Models that are based on the Gaussian Mixture Models can be used to classify the task into subtasks to identify the intent of the surgeons. This would definitely help in filtering out the data due to the Gaussian functions being used and are generally used to model human behaviors. However, the difficulty would be to incorporate the time series information which is very important to this classification. A possible solution could be to include derivatives of the motion data for the time series information.

8.2.2 Personalization of automated camera motion

During traditional laparoscopic surgeries, surgeon's prefer to work with specific surgeons and staff that over get to know the preferences of surgeons viewpoint. All surgeon's have different preferences and the approach to automation presented in this thesis can be definitely used to personalize the algorithm to generate camera trajectories based on the surgeon's preferences. If only the surgeon's data is used to train the inverse reinforcement learning algorithm, the weights obtained would reflect the preference of the surgeon.

8.2.3 Skill assessment and teaching

Automation of camera motion can also be used to assess skills of a surgeon and also to teach lesser experienced surgeons. If the algorithm is trained using data from several expert surgeons, it will extract the information and generate trajectories similar to the expert's trajectories. These trajectories can be used as a baseline to evaluate the expertise of surgeon's skills using the similarity metric presented in this thesis. Moreover, a work similar to [81] can be performed to differentiate between novice and expert surgeons. Also it would be a very useful study to evaluate the cognitive load on the surgeons with and without the automation of the endoscope using some metrics similar to the ones used in the study [82].

Furthermore, it can be used to teach the resident surgeon's the importance of camera viewpoints and what to look for during certain scenarios of the surgery. This would be very similar to how a senior surgeon teaches a junior resident but in this case, the resident could learn by themselves. This would help in providing a uniform level of education in the medical field, potentially all over the world.

8.2.4 Implementation on hardware

In addition to improving the accuracy of the algorithm, the algorithm can be extended to be implemented on the hardware. The way this algorithm was designed, the implementation of the hardware should be straight forward. However, the PSMs should be registered with respect to the endoscope. That is, the position and orientation of the remote center of motions of the PSM should be known either to a common world or the RCM of the ECM. Once the PSMs are registered, the states for the algorithm can be calculated using the parameters of the endoscope camera (like focal length, field of view, etc.). These states can be given to the corresponding policies to get the automated trajectories of the endoscope. Though to perform the task, the motion of the MTM need to be mapped to the motion of the PSM in the ECM frame. In other words, if the MTM handle is moved right, the instrument tip should be moving right in the image frame for the exercise to be intuitive to the user. This involves calculating with the inverse kinematics of the PSM which can get complicated with the joint limits and singularities present in the wristed instruments.

8.2.5 Extensions of the work

Furthermore, the approach provided here can be applied for two or arms which is the case during surgery. It is easier to extend the approach for the same task for two hands. Though some of the features will need to be added to include the interaction of the two arms and consider the relative positions of the two arms with respect to each other and the endoscope. The user study would need to be performed again to collect the data with two arms rather than one. This is one of the major downsides of using inverse reinforcement learning that user data in form of trajectories are

CONCLUSION AND FUTURE WORK

needed. Therefore, it will be really useful and important to have open data sets to advance the research in this field.

In addition, it would be extremely valuable to have data sets from expert surgeons that perform various tasks. These datasets would be similar to the JIGSAW data set [19] but with the camera allowed to move when the instruments are moving. The same virtual reality approach can be used. Also, it would be useful to track the position of the surgeon's head which could be mapped to the insertion (distance from the screens or similar) of the endoscope rather than having a foot pedal for it. This would help reduce the bias against changing the insertion due to it being a separate operation.

Majority of the time during the thesis was spent in developing the simulated environment for the user study. However there are several simulators out there that provide various tasks such as needle passing, suturing, etc. that are used to train surgeon's. [83] provides an example of such a simulator that is used by surgeon's to get accustomed to the daVinci surgical robot. Though it is not open source, if the kinematic data from the simulator can be obtained a user study can be designed to have expert surgeons use it while wearing virtual reality headsets to collect data to be used. This would allow to collect data for a lot of tasks simultaneously and would be very time efficient.

If the data for several tasks is available, the approach taken in this thesis can be directly applied to automate the endoscope for all of the surgical tasks. Feature sets would need to be defined according to the tasks but the algorithms could directly be used. These tasks then can be combined to achieve automation for the whole surgical procedure.

Bibliography

- [1] Vierra, MD, Mark. "Minimally invasive surgery." *Annual review of medicine* 46, no. 1 (1995): 147-158.
- [2] Lanfranco, Anthony R., Andres E. Castellanos, Jaydev P. Desai, and William C. Meyers. "Robotic surgery: a current perspective." *Annals of surgery* 239, no. 1 (2004): 14.
- [3] Gomes, Paula. "Surgical robotics: Reviewing the past, analysing the present, imagining the future." *Robotics and Computer-Integrated Manufacturing* 27, no. 2 (2011): 261-266.
- [4] Shah, Rachit D., Alex Cao, Lavie Golenberg, R. Darin Ellis, Gregory W. Auner, Abhilash K. Pandya, and Michael D. Klein. "Performance of basic manipulation and intracorporeal suturing tasks in a robotic surgical system: single-versus dual-monitor views." *Surgical endoscopy* 23, no. 4 (2009): 727-733.
- [5] Ritter, E.M. and Scott, D.J., 2007. Design of a proficiency-based skills training curriculum for the fundamentals of laparoscopic surgery. *Surgical innovation*, 14(2), pp.107-112.
- [6] Gers, F.A., Schmidhuber, J. and Cummins, F., 1999. Learning to forget: Continual prediction with LSTM.
- [7] Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. Vol. 1. No. 1. Cambridge: MIT press, 1998.
- [8] Leonard, S., Wu, K.L., Kim, Y., Krieger, A. and Kim, P.C., 2014. Smart tissue anastomosis robot (STAR): A vision-guided robotics system for laparoscopic suturing. *IEEE Transactions on Biomedical Engineering*, 61(4), pp.1305-1317.
- [9] Shademan, A., Decker, R.S., Opfermann, J.D., Leonard, S., Krieger, A. and Kim, P.C., 2016. Supervised autonomous robotic soft tissue surgery. *Science translational medicine*, 8(337), pp.337ra64-337ra64.
- [10] Opfermann, J.D., Leonard, S., Decker, R.S., Uebele, N.A., Bayne, C.E., Joshi, A.S. and Krieger, A., 2017, September. Semi-autonomous electrosurgery for tumor resection using a multi-degree of freedom electrosurgical tool and visual

CONCLUSION AND FUTURE WORK

- servoing. In Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on (pp. 3653-3660). IEEE.
- [11] Yip, M. and Das, N., 2017. Robot autonomy for surgery. arXiv preprint arXiv:1707.03080.
- [12] Moustris, G.P., Hiridis, S.C., Deliparaschos, K.M. and Konstantinidis, K.M., 2011. Evolution of autonomous and semiautonomous robotic surgical systems: a review of the literature. *The international journal of medical robotics and computer assisted surgery*, 7(4), pp.375-392.
- [13] Murali, A., Sen, S., Kehoe, B., Garg, A., McFarland, S., Patil, S., Boyd, W.D., Lim, S., Abbeel, P. and Goldberg, K., 2015, May. Learning by observation for surgical subtasks: Multilateral cutting of 3d viscoelastic and 2d orthotropic tissue phantoms. In Robotics and Automation (ICRA), 2015 IEEE International Conference on (pp. 1202-1209). IEEE.
- [14] Thananjeyan, B., Garg, A., Krishnan, S., Chen, C., Miller, L. and Goldberg, K., 2017, May. Multilateral surgical pattern cutting in 2d orthotropic gauze with deep reinforcement learning policies for tensioning. In Robotics and Automation (ICRA), 2017 IEEE International Conference on (pp. 2371-2378). IEEE.
- [15] Rosenstein, Michael T., Andrew H. Fagg, and Roderic A. Grupen. "Robot learning with predictions of operator intent." *Papers from the 2004 AAAI Fall Symposium on The Intersection of Cognitive Science and Robotics: From Interfaces to Intelligence*, Technical Report FS-04-05. 2004.
- [16] Panzner, M. and Cimiano, P., 2016, August. Comparing hidden markov models and long short term memory neural networks for learning action representations. In International Workshop on Machine Learning, Optimization and Big Data (pp. 94-105). Springer, Cham.
- [17] Padoy, N. and Hager, G.D., 2011, May. Human-machine collaborative surgery using learned models. In Robotics and Automation (ICRA), 2011 IEEE International Conference on (pp. 5285-5292). IEEE.
- [18] Lin, H.C., Shafran, I., Murphy, T.E., Okamura, A.M., Yuh, D.D. and Hager, G.D., 2005, October. Automatic detection and segmentation of robot-assisted surgical motions. In International Conference on Medical Image Computing and Computer-Assisted Intervention (pp. 802-810). Springer, Berlin, Heidelberg.
- [19] Gao, Y., Vedula, S.S., Reiley, C.E., Ahmidi, N., Varadarajan, B., Lin, H.C., Tao, L., Zappella, L., Bjar, B., Yuh, D.D. and Chen, C.C.G., 2014. JHU-ISI gesture and skill assessment working set (JIGSAWS): A surgical activity dataset for human motion modeling. In MICCAI Workshop: M2CAI (Vol. 3, p. 3).

CONCLUSION AND FUTURE WORK

- [20] Ahmidi, N., Tao, L., Sefati, S., Gao, Y., Lea, C., Haro, B.B., Zappella, L., Khudanpur, S., Vidal, R. and Hager, G.D., 2017. A dataset and benchmarks for segmentation and recognition of gestures in robotic surgery. *IEEE Transactions on Biomedical Engineering*, 64(9), pp.2025-2041.
- [21] Radermacher, K., Anton, M., Rau, G., Boeckmann, W., Jakse, G. and Staudte, H.W., 1997. Development of an automatic surgical holding system based on ergonomic analysis. In *CVRMed-MRCAS'97* (pp. 737-746). Springer, Berlin, Heidelberg.
- [22] Schurr, M.O., Arezzo, A., Neisius, B., Rininsland, H., Hilzinger, H.U., Dorn, J., Roth, K. and Buess, G.F., 1999. Trocar and instrument positioning system TISKA. *Surgical endoscopy*, 13(5), pp.528-531.
- [23] Bihlmaier, A. and Bihlmaier, 2016. Learning dynamic spatial relations. Wiesbaden: Springer Vieweg.
- [24] Nathan, C.A.O., Chakradeo, V., Malhotra, K., D'Agostino, H. and Patwardhan, R., 2006. The voice-controlled robotic assist scope holder AESOP for the endoscopic approach to the sella. *Skull Base*, 16(3), p.123.
- [25] Ali, S.M., Reisner, L.A., King, B., Cao, A., Auner, G., Klein, M. and Pandya, A.K., 2008. Eye gaze tracking for endoscopic camera positioning: an application of a hardware/software interface developed to automate Aesop. *Studies in health technology and informatics*, 132, pp.4-7.
- [26] Lee, C., Wang, Y.F., Uecker, D.R. and Wang, Y., 1994, September. Image analysis for automated tracking in robot-assisted endoscopic surgery. In *Proceedings of 12th International Conference on Pattern Recognition* (pp. 88-92). IEEE.
- [27] Wei, G.Q., Arbter, K. and Hirzinger, G., 1997. Real-time visual servoing for laparoscopic surgery. Controlling robot motion with color image segmentation. *IEEE Engineering in Medicine and Biology Magazine*, 16(1), pp.40-45.
- [28] Omote, K., Feussner, H., Ungeheuer, A., Arbter, K., Wei, G.Q., Siewert, J.R. and Hirzinger, G., 1999. Self-guided robotic camera control for laparoscopic surgery compared with human camera control. *The American journal of surgery*, 177(4), pp.321-324.
- [29] Voros, S., Haber, G.P., Menudet, J.F., Long, J.A. and Cinquin, P., 2010. ViKY robotic scope holder: Initial clinical experience and preliminary results using instrument tracking. *IEEE/ASME transactions on mechatronics*, 15(6), pp.879-886.

- [30] Voros, S., Long, J.A. and Cinquin, P., 2007. Automatic detection of instruments in laparoscopic images: A first step towards high-level command of robotic endoscopic holders. *The International Journal of Robotics Research*, 26(11-12), pp.1173-1190.
- [31] Kwon, D.S., Ko, S.Y. and Kim, J., 2008. Intelligent laparoscopic assistant robot through surgery task model: how to give intelligence to medical robots. In *Medical Robotics*. InTech.
- [32] Mudunuri, A.V., 2010. Autonomous camera control system for surgical robots. Wayne State University.
- [33] Eslamian, S., Reisner, L.A., King, B.W. and Pandya, A.K., An Autonomous Camera System using the da Vinci Research Kit.
- [34] Eslamian, S., Reisner, L.A., King, B.W. and Pandya, A.K., 2016, April. Towards the Implementation of an Autonomous Camera Algorithm on the da Vinci Platform. In *MMVR* (pp. 118-123).
- [35] Azizian, M., Khoshnam, M., Najmaei, N. and Patel, R.V., 2014. Visual servoing in medical robotics: a survey. Part I: endoscopic and direct vision imaging techniques and applications. *The international journal of medical robotics and computer assisted surgery*, 10(3), pp.263-274.
- [36] Bani, M.J., Autonomous Camera Movement for Robotic-Assisted Surgery: A Survey. *International Journal of Advanced Engineering, Management and Science*, 3(8).
- [37] Pandya, A., Reisner, L., King, B., Lucas, N., Composto, A., Klein, M. and Ellis, R., 2014. A review of camera viewpoint automation in robotic and laparoscopic surgery. *Robotics*, 3(3), pp.310-329.
- [38] Ng, A.Y. and Russell, S.J., 2000, June. Algorithms for inverse reinforcement learning. In *Icml* (pp. 663-670).
- [39] Ramachandran, D. and Amir, E., 2007. Bayesian inverse reinforcement learning. *Urbana*, 51(61801), pp.1-4.
- [40] Lopes, M., Melo, F. and Montesano, L., 2009, September. Active learning for reward estimation in inverse reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (pp. 31-46). Springer, Berlin, Heidelberg.
- [41] Vroman, M.C., 2014. Maximum likelihood inverse reinforcement learning (Doctoral dissertation, Rutgers University-Graduate School-New Brunswick).

CONCLUSION AND FUTURE WORK

- [42] Argall, B.D., Chernova, S., Veloso, M. and Browning, B., 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5), pp.469-483.
- [43] Arora, S. and Doshi, P., 2018. A Survey of Inverse Reinforcement Learning: Challenges, Methods and Progress. *arXiv preprint arXiv:1806.06877*.
- [44] Ziebart, B.D., Maas, A.L., Bagnell, J.A. and Dey, A.K., 2008, July. Maximum Entropy Inverse Reinforcement Learning. In *AAAI* (Vol. 8, pp. 1433-1438).
- [45] Hadfield-Menell, D., Russell, S.J., Abbeel, P. and Dragan, A., 2016. Cooperative inverse reinforcement learning. In *Advances in neural information processing systems* (pp. 3909-3917).
- [46] Levine, S. and Koltun, V., 2012. Continuous inverse optimal control with locally optimal examples. *arXiv preprint arXiv:1206.4617*.
- [47] Bishop, C.M., 1995. *Neural networks for pattern recognition*. Oxford university press.
- [48] Grossberg, S., 2013. Recurrent neural networks. *Scholarpedia*, 8(2), p.1888.
- [49] Wang, J.S., *Recurrent Neural Nets*.
- [50] Bertolami, R., Bunke, H., Fernandez, S., Graves, A., Liwicki, M. and Schmidhuber, J., 2009. A Novel Connectionist System for Improved Unconstrained Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5).
- [51] Sak, H., Senior, A. and Beaufays, F., 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Fifteenth annual conference of the international speech communication association*.
- [52] Sak, H., Senior, A. and Beaufays, F., 2014. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*.
- [53] Funahashi, K.I. and Nakamura, Y., 1993. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural networks*, 6(6), pp.801-806.
- [54] Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural computation*, 9(8), pp.1735-1780.
- [55] Olah, C., 2015. Understanding lstm networks, 2015. URL <http://colah.github.io/posts/2015-08-Understanding-LSTMs>.

CONCLUSION AND FUTURE WORK

- [56] Duan, Y., Chen, X., Houthoofd, R., Schulman, J. and Abbeel, P., 2016, June. Benchmarking deep reinforcement learning for continuous control. In International Conference on Machine Learning (pp. 1329-1338).
- [57] Boone, G., 1997, April. Efficient reinforcement learning: Model-based acrobot control. In Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on (Vol. 1, pp. 229-234). IEEE.
- [58] Bou-Ammar, H., Voos, H. and Ertel, W., 2010, September. Controller design for quadrotor uavs using reinforcement learning. In Control Applications (CCA), 2010 IEEE International Conference on (pp. 2130-2135). IEEE.
- [59] Peters, J., Vijayakumar, S. and Schaal, S., 2003, September. Reinforcement learning for humanoid robotics. In Proceedings of the third IEEE-RAS international conference on humanoid robots (pp. 1-20).
- [60] Shalev-Shwartz, S., Shammah, S. and Shashua, A., 2016. Safe, multi-agent, reinforcement learning for autonomous driving. arXiv preprint arXiv:1610.03295.
- [61] Kormushev, P., Calinon, S. and Caldwell, D.G., 2013. Reinforcement learning in robotics: Applications and real-world challenges. *Robotics*, 2(3), pp.122-148.
- [62] White III, C.C. and White, D.J., 1989. Markov decision processes. *European Journal of Operational Research*, 39(1), pp.1-16.
- [63] Kazanzides, P., Chen, Z., Deguet, A., Fischer, G.S., Taylor, R.H. and DiMaio, S.P., 2014, May. An open-source research kit for the da Vinci Surgical System. In Robotics and Automation (ICRA), 2014 IEEE International Conference on (pp. 6434-6439). IEEE.
- [64] Deguet, A., Kumar, R., Taylor, R. and Kazanzides, P., 2008, September. The cisst libraries for computer assisted intervention systems. In MICCAI Workshop on Systems and Arch. for Computer Assisted Interventions, Midas Journal (Vol. 71).
- [65] Vagvolgyi, B., DiMaio, S., Deguet, A., Kazanzides, P., Kumar, R., Hasser, C. and Taylor, R., 2008, September. The surgical assistant workstation. In Proc MICCAI Workshop: Systems and Architectures for Computer Assisted Interventions (pp. 1-8).
- [66] Chen, Z., Deguet, A., Taylor, R., DiMaio, S., Fischer, G. and Kazanzides, P., 2013, September. An open-source hardware and software platform for telesurgical robotics research. In Proceedings of the MICCAI Workshop on Systems and Architecture for Computer Assisted Interventions, Nagoya, Japan (Vol. 2226).

CONCLUSION AND FUTURE WORK

- [67] Chen, Z., Deguet, A., Taylor, R.H. and Kazanzides, P., 2017, April. Software architecture of the da Vinci Research Kit. In *Robotic Computing (IRC)*, IEEE International Conference on (pp. 180-187). IEEE.
- [68] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., ... & Ng, A. Y. (2009, May). ROS: an open-source Robot Operating System. In *ICRA workshop on open source software* (Vol. 3, No. 3.2, p. 5).
- [69] Koenig, N., & Howard, A. (2004, September). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004.(IROS 2004)*. Proceedings. 2004 IEEE/RSJ
- [70] 2018. GitHub repository, <https://github.com/OpenHMD>.
- [71] 2018. Github repository, https://github.com/h3ct0r/openhmd_ros
- [72] LaValle, S.M., Yershova, A., Katsev, M. and Antonov, M., 2014, May. Head tracking for the Oculus Rift. In *Robotics and Automation (ICRA)*, 2014 IEEE International Conference on (pp. 187-194). IEEE.
- [73] Conti, F., 2003. The CHAI libraries (No. LSRO2-CONF-2003-003).
- [74] Conti, Francois, D. Morris, F. Barbagli, and C. Sewell. "CHAI 3D." Online: <http://www.chai3d.org> (2006).
- [75] Coumans, E., 2010. Bullet physics engine. Open Source Software: <http://bulletphysics.org>, 1, p.3.
- [76] Munawar A., 2018. GitHub repository, https://github.com/WPI-AIM/chai_env
- [77] Kingma, D.P. and Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980).
- [78] Bottou, L., 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010* (pp. 177-186). Physica-Verlag HD.
- [79] Chollet, F., 2018. Keras: The python deep learning library. Astrophysics Source Code Library.
- [80] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. and Kudlur, M., 2016, November. Tensorflow: a system for large-scale machine learning. In *OSDI* (Vol. 16, pp. 265-283).
- [81] Law, B., Atkins, M.S., Kirkpatrick, A.E. and Lomax, A.J., 2004, March. Eye gaze patterns differentiate novice and experts in a virtual laparoscopic surgery training environment. In *Proceedings of the 2004 symposium on Eye tracking research & applications* (pp. 41-48). ACM.

CONCLUSION AND FUTURE WORK

- [82] Guru, K.A., Esfahani, E.T., Raza, S.J., Bhat, R., Wang, K., Hammond, Y., Wilding, G., Peabody, J.O. and Chowriappa, A.J., 2015. Cognitive skills assessment during robotassisted surgery: separating the wheat from the chaff. *BJU international*, 115(1), pp.166-174.
- [83] Rogula, T., Acquafresca, P.A. and Bazan, M., 2015. Training and credentialing in robotic surgery. In *Essentials of Robotic Surgery* (pp. 13-26). Springer, Cham.