Teleoperation, Human Intent Prediction and Imitation Learning Methods for Collaborative Robots

by

Sanket Gaurav
B.Tech (Sikkim Manipal University) 2014
M.S. (University of Illinois at Chicago) 2017

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2021

Chicago, Illinois

Defense Committee:
Brian Ziebart, Chair and Advisor
Ajay Kshemkalyani
Wei Tang
Milos Zefran, Electrical and Computer Engineering
Venugopal Vasudevan, Procter & Gamble

For my Big Brother Rahul Gaurav.

ACKNOWLEDGMENTS

I have the great pleasure to thank my advisor - Professor Brian Ziebart - to help and guide me in my interests in machine learning and robotics and brought the best of me for the last six years (two years of MS and four years of Ph.D.). I would like to thank Professor Ajay Kshemkalyani, Professor Wei Tang, Professor Milos Zefran, and Dr. Venugopal Vasudevan for their effort and time to participate in my Ph.D. thesis committee.

First, I would like to thank my fellow co-authors of the accepted HUMANOIDS 2019 paper and submitted CoRL 2021: Amey Barapatre, Geroge Maratos, Tejas Sarma, and Zainab Al-Qurashi, Aaron Crookes, Vignesh Narayanaswamy, Harish Venkataraman, Matthew Barker, and Venugopal Vasudevan. All of them helped in the creation of the robotic system and experiments for the research work. At Procter & Gamble, I would like to thank my colleagues Henrique Aveiro, Noah Kirst, Matthew Bauer, Su Chang, Robert Baker, and other members of DS/AI for their help in data collection and other discussions related to the Imitation Learning project.

I would also thank all my lab members in the Purposeful Prediction lab for their continuous support and help. I thank all the participants who participated in our training and testing data collection for HUMANOIDS 2019 and AAMAS 2019 paper. Thank you to Mathew Monfort for sharing his linear quadratic regulator for inverse optimal control code with us for modification for AAMAS 2019 paper. Also, I thank Sima Behopour, Kaiser Asif, Omid Memarrast, and Rizal Fathony for having interesting discussions in the lab.

I extend my gratitude towards my parents for giving me the freedom and support to achieve my dreams and aspirations. Next, I would like to thank my big brother, Rahul Gaurav for helping me dream big and pushing my limits to achieve them. Finally, I thank my family and friends for helping and supporting me in this journey.

Lastly, I would like to thank the Future of Life Institute (futureoflife.org) FLI-RFP-AI1 program, grant #2016-158710, and NSF grant #1652530 for supporting this work. I would also thank Procter & Gamble to support my Ph.D. thesis work for the last one and a half years.

SG

Contribution of Authors

Chapter 1 gives an introduction to the thesis and is solely written by me. Chapter 2 is based on our research paper titled Deep Correspondence Learning for Effective Robotic Teleoperation using Virtual Reality (1) published at IEEE International Conference on Humanoid Robots (Humanoids) 2022. My co-authors Zainab, Amey, George, and Tejas helped with data collection and experimentation for the paper and my advisor Prof. Brian Ziebart supervise the paper and helped with manuscript correction. I was the primary author. Chapter 3 is obtained from the research paper titled Discriminatively Learning Inverse Optimal Control Models for Predicting Human Intentions (2) published at ACM International Conference on Autonomous Agent and Multi-Agent Systems (AAMAS) 2019. My advisor, Prof. Brian Ziebart guided this paper and helped with manuscript correction. I was the primary author of this paper as well.

SG

TABLE OF CONTENTS

CHAPTER			PAGE
1	**INTRODUCTION**		1
	1.1	Outline of the document	6
2	**DEEP CORRESPONDENCE LEARNING FOR EFFECTIVE ROBOTIC TELEOPERATION USING VIRTUAL REALITY**		9
	2.1	Introduction	9
	2.2	Background and Related Work	12
	2.2.1	Correspondence Learning in Virtual Reality	12
	2.2.2	Baxter Robot	14
	2.2.3	Deep Learning Architecture	14
	2.2.4	Loss Function	16
	2.2.4.1	Mean Squared Error	16
	2.2.4.2	Cosine Similarity	16
	2.3	Approach	16
	2.3.1	Visualizing Microsoft Kinect depth data in Virtual Reality	17
	2.3.2	Robotic Teleoperation using Virtual Reality	17
	2.3.3	Deep Correspondence Learning Architecture	18
	2.4	Experiment	19
	2.4.1	Hardware Setup	19
	2.4.2	Data Collection	20
	2.4.3	Data Pre-processing	20
	2.4.4	Training of Baselines	21
	2.4.5	Training of Deep Learning Models	21
	2.4.6	Experimental Setup	22
	2.4.6.1	Offline Experiment	22
	2.4.6.2	Real-time Robotic Teleoperation	24
	2.5	Results and Discussion	25
	2.6	Conclusion and Future Work	29
	2.7	Acknowledgment	30
3	**DISCRIMINATIVELY LEARNING INVERSE OPTIMAL CONTROL MODELS FOR PREDICTING HUMAN INTENTIONS**		31
	3.1	Introduction	31
	3.2	Background and Related Work	33
	3.2.1	Decision Processes and Goal Prediction	33
	3.2.2	Existing Goal Prediction Methods	35
	3.2.3	Maximum Entropy Inverse Optimal Control	37

CHAPTER			PAGE
	3.2.4	Inverse Linear-Quadratic-Regulation	40
	3.3	Approach	41
	3.3.1	Goal Likelihood Maximization Formulation	42
	3.3.2	Extension to Linear-Quadratic Regulation	45
	3.3.3	Complexity Analysis	47
	3.4	Experimental Setup	47
	3.4.1	Goal Pointing Task Data	47
	3.4.2	Cornell Activity Dataset (CAD-120)	50
	3.4.3	Estimating the Reward Parameters	50
	3.4.4	Goal Prediction via Inverse LQR	51
	3.4.5	Prior Distribution	53
	3.4.6	Baselines	54
	3.4.7	Evaluation Metrics	54
	3.5	Results and Discussion	55
	3.6	Conclusion and Future Work	59
	3.7	Acknowledgment	60
4	**ROBOT LEARNING TO MOP LIKE HUMANS USING VIDEO DEMONSTRATIONS**		61
	4.1	Introduction	61
	4.2	Approach	64
	4.2.1	Method 1: Tracking and Inverse Kinematics Approach	64
	4.2.2	Method 2: Time Contrastive Network and Reinforcement Learning Approach	67
	4.2.2.1	Time Contrastive Network	67
	4.2.2.2	Reinforcement Learning to learn mopping behavior	68
	4.2.2.3	Reinforcement Learning (PILQR) Details	72
	4.3	Experiment	74
	4.3.1	Hardware Used	74
	4.3.2	Data Collection	74
	4.3.3	Data Pre-processing and Video Alignment	74
	4.3.4	Training TCN model	78
	4.3.5	Simulation of UR10e with mop	79
	4.3.6	Learning Human mopping to the robotic arm using RL	80
	4.3.7	Evaluation of the Robot mopping system	81
	4.3.7.1	Cosine Similarity	81
	4.3.7.2	Euclidean Distance Loss	82
	4.4	Result	82
	4.5	Related Work	83
	4.6	Conclusion and Future Work	85
	4.7	Acknowledgment	86

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

5 CONCLUSION AND FUTURE WORK 87

APPENDICES 89

Appendix A 90

CITED LITERATURE 93

VITA 107

LIST OF FIGURES

FIGURE **PAGE**

1 How can we effectively teleoperate a robot in virtual reality using deep correspondence learning? 3

2 Can we predict human intention more quickly based on the partial trajectory traveled by maximizing the goal likelihood for Inverse Optimal Control? 4

3 Can a robot learn a day-to-day task (for example, mopping the floor) from a human demonstration provided in the form of a video? 5

4 (a) Visualization of the Baxter robot's workspace captured from a depth camera (Microsoft Kinect) and displayed on a virtual reality headset (HTC Vive); (b) demonstrator holding HTC Vive controller slowly moves his hand from neutral position; (c) trainer moves robot hand manually to follow human trajectory; and (d) demonstrator and trainer with Baxter reaching the same configuration. 10

5 Training of the deep neural network to learn the correspondence between the Baxter robot joint angles and the human poses where input 14 represents the position and rotation data of HTC Vive controller placed in two respective hand of the human demonstrator and output 7 represents the seven joint angles of one of the Baxter robot's arm. 15

6 (a) Pick and place experiment setup in virtual reality, (b) Teleoperator reaching the object (white box) kept at position A in virtual reality, (c) Teleoperator picking up the white box from position A and moving to position B in virtual reality, (d) Baxter teleoperated to reach the object (white box) kept at position A, (e) Pick and place experiment setup in reality, (f) Baxter teleoperated to reach the object (white box) kept at position A, (g) Baxter teleoperated to pick up the white box from position A and moving to position B, (h) Baxter teleoperated to place the box at position B. 23

7 (a)-(c) Robot end-effector positions (x_t, y_t, z_t) plotted for Ground Truth vs Linear Regression vs Deep Network; (d)-(g) Robot end-effector rotational angles (x_r, y_r, z_r, w_r) plotted for Ground Truth vs Linear Regression vs Deep Network. 26

LIST OF FIGURES (Continued)

FIGURE		PAGE
8	A partial trajectory ($S_{1:t_i}$) and distribution for two goals in the space at a trajectory point S_t. Inverse LQR with the trained reward parameters using the algorithm 1 are used to calculate the distributions for both goals.	46
9	The steps of a task in our testing sequence from a pointing dataset (3) include starting from the robot's neutral position (a) and then teleoperating the arm of the robot (b) to the goal location (c) at which point confirmation is displayed on the robot's screen (d).	48
10	(a) Plot showing comparison of logloss by our goal likelihood maximization method to the trajectory likelihood maximization model on goal pointing task data; (b) Change of probability distribution over goals across a trajectory of reaching goal #3 using trajectory likelihood maximization model; (c) Change of probability distribution over goals across a trajectory of reaching goal #3 using goal likelihood maximization method; (d) Plot showing comparison of logloss by goal likelihood to trajectory likelihood on CAD-120.	52
11	Robot learning to mop from human mopping video demonstration. . . .	62
12	Time Contrastive Network Architecture	69
13	Frames extraction from videos for TCN	70
14	(a) Saw tooth straight forward/angle back, (b)Forward + left 90°, (c) Forward + right 90°, (d) Scrubbing between point 1 and 2	75
15	(a) Blue sheet to signal the demonstrator to start performing task (b) Temporal arrangement of frames before synchronization (c) Temporal arrangement of frames after synchronization.	77

LIST OF TABLES

TABLE		PAGE
I	RESULTS (TEST LOSS IN RMSE) OF HTC VIVE CONTROLLER TO BAXTER CORRESPONDENCE MODELS	25
II	EUCLIDEAN DISTANCE MEASURE FROM GROUND TRUTH BAXTER'S END-EFFECTOR POSITION TO PREDICTED BAXTER'S END-EFFECTOR	25
III	COSINE SIMILARITY RESULT	27
IV	SUCCESS RATE OF PICK & PLACE EXPERIMENT	27
V	A COMPARISON OF THE TRAJECTORY LIKELIHOOD MODEL, THE GOAL LIKELIHOOD MODEL, AND THE NEAREST GOAL BASELINE FOR THE GOAL POINTING TASK DATASET EVALUATED USING THE ACCURACY, MACRO PRECISION, AND MACRO RECALL GIVEN VARIOUS FRACTIONS OF THE TRAJECTORY.	55
VI	A COMPARISON OF THE TRAJECTORY LIKELIHOOD MODEL, THE GOAL LIKELIHOOD MODEL, AND THE ATCRF MODEL FOR THE CAD-120 DATASET EVALUATED USING ACCURACY, MACRO PRECISION, AND MICRO PRECISION GIVEN VARIOUS FRACTIONS OF THE TRAJECTORY.	56
VII	EVALUATION RESULT OF THE TWO METHODS	83

LIST OF ABBREVIATIONS

LQR	Linear Quadratic Regulator
UIC	University of Illinois at Chicago
IOC	Inverse Optimal Control
DOF	Degree of Freedom
ROS	Robot Operating System
IRL	Inverse Reinforcement Learning
RL	Reinforcement Learning
VR	Virtual Reality
TCN	Time Contrastive Networks

SUMMARY

Some limitations and challenges prevent robots from being accepted widely as human peers. This thesis studies prediction and learning methods to understand human intentions and address challenges posed to collaborative robots related to translating between human and robotic behavior. The first topic discusses the correspondence learning problem of estimating a mapping of human embodiment to robot-joint configuration for robotic teleoperation using virtual reality. The second topic seeks to enable robots to more accurately predict human intentions from partial trajectories so that the robot can plan complementary activities. Finally, the research extends to learn a daily human activity (mopping the floor) from human demonstration videos to a robotic arm.

By projecting into a 3-D workspace, robotic teleoperation using virtual reality allows for a more intuitive method of control for the operator than using a 2-D view from the robot's visual sensors. This chapter investigates a setup that places the teleoperator in a virtual representation of the robot's environment and develops a deep learning based architecture modeling the correspondence between the operator's movements in the virtual space and joint angles for a humanoid robot using data collected from a series of demonstrations. We evaluate the correspondence model's performance in a pick-and-place teleoperation experiment.

More accurately inferring human intentions/goals can help robots complete collaborative human-robot tasks more safely and efficiently. Bayesian reasoning has become a popular approach for predicting the intention or goal of a partial sequence of actions/controls using a trajectory likelihood model. However, the mismatch between the training objective for these models (maximizing trajectory likelihood) and the application objective (maximizing intention likelihood) can be detrimental. In this chapter, we seek to improve the goal prediction of maximum entropy inverse reinforcement learning (MaxEnt IRL) models by training to maximize goal likelihood. We demonstrate the benefits of our method on pointing task goal prediction with multiple possible goals and predicting goal based activities in the Cornell Activity Dataset (CAD-120).

Though mopping the floor is a mundane and tedious daily task, enabling robots to perform it comparably to humans remains a challenge. Hand-coding desired mopping behaviors for variable surfaces and situations is particularly difficult. In this chapter, we develop a robotic system for mopping the floor by mimicking the human behavior demonstrated in videos. Our approach builds upon the recent successes of imitation learning of other capabilities from human video demonstration (e.g., pouring tasks (4)). Our first proposed robotic system uses traditional computer vision techniques for tracking and inverse kinematics. Our second system comprises advanced computer vision techniques, Time Contrastive Network (TCN), and reinforcement learning. From these, we devise a reward function for the mopping task. We use a Universal 10e robotic arm attached with a mop to perform the mopping task and a first-person camera attached on top of the robotic arm to provide feedback for robot learning.

CHAPTER 1

INTRODUCTION

Today robots are becoming an integral part of human life. From delivery drones to self-driving cars to old age assisting robots to industrial robots, these different service robots are helping in our day-to-day life. However, there are time and resource limitations of robots that must be addressed before they are integrated into daily operations of human life. This thesis studies time and resource efficient ways to provide service robots with a human task domain understanding, so they can be effective collaborators and engender trust with their human counterparts. It aims to answer following questions:

1. How can we effectively teleoperate a robot in virtual reality (VR) using deep correspondence learning?
2. Can we predict human intention more quickly based on the partial trajectory traveled by maximizing the goal likelihood for Inverse Optimal Control?
3. Can a robot learn a day-to-day task (for example, mopping the floor) from a human demonstration provided in the form of a video?

The Ph.D. thesis addresses these time and resource limitations of collaborative robots. The first chapter discusses how a robot can effectively learn in a synthetic environment (VR) so that the velocity of learning is not constrained by collecting real-world data sets. The next chapter aims to gain understanding of human intent so the robot can extrapolate (vs mimic) what the human does or wants. The goal of the final chapter is to simplify the collection of real-world data so the robot can learn themes, variations, and outliers of human behavior by example from YouTube videos. The final chapter also shows the viability and challenges in translating this 4-dimensional data (3-D plus time) pertaining to human tasks into robotic test methods. Lastly, it discusses future research needed to ensure that these methods are 'consumer grade'.

Many tasks are difficult and life-threatening to humans (for example, bomb diffusion, conducting experiments with hazardous materials in chemical labs, etc.). Robotic teleoperation, teleoperating (controlling) a robot from a distance is a possible way to handle the beforementioned problems (5; 6; 7; 8; 9; 10). The first chapter of the thesis investigates the correspondence learning problem (mapping of human embodiment to robot embodiment) for robotic teleoperation using virtual reality. In this chapter, the traditional 2D input sensor used for teleoperation is replaced with a 3D (virtual reality) system (11; 12; 13; 14; 15; 16) to provide a depth perspective to the teleoperators. The consumer-grade virtual reality system like HTC Vive can track the HTC Vive components at 60Hz which is equivalent to real-time tracking. A Baxter robot, HTC Vive, and a Kinect camera attached to the top of the robot is used to visualize the robot space to the virtual reality system. By introducing virtual reality (11; 12; 13; 14; 15; 16), the robotic teleoperation system encounters sparse tracking points for the human to robot correspondence learning model. The traditional 2D system (9; 17; 18; 3; 19; 20) maps a human body to a skeleton comprised of 15 joint positions. However, the HTC Vive only consists of 3 positions (two – each human wrist holding an HTC Vive Controller– and head position). Thus, this transfer of high dimensional human end-effector (HTC Vive Controller positions) to lower-

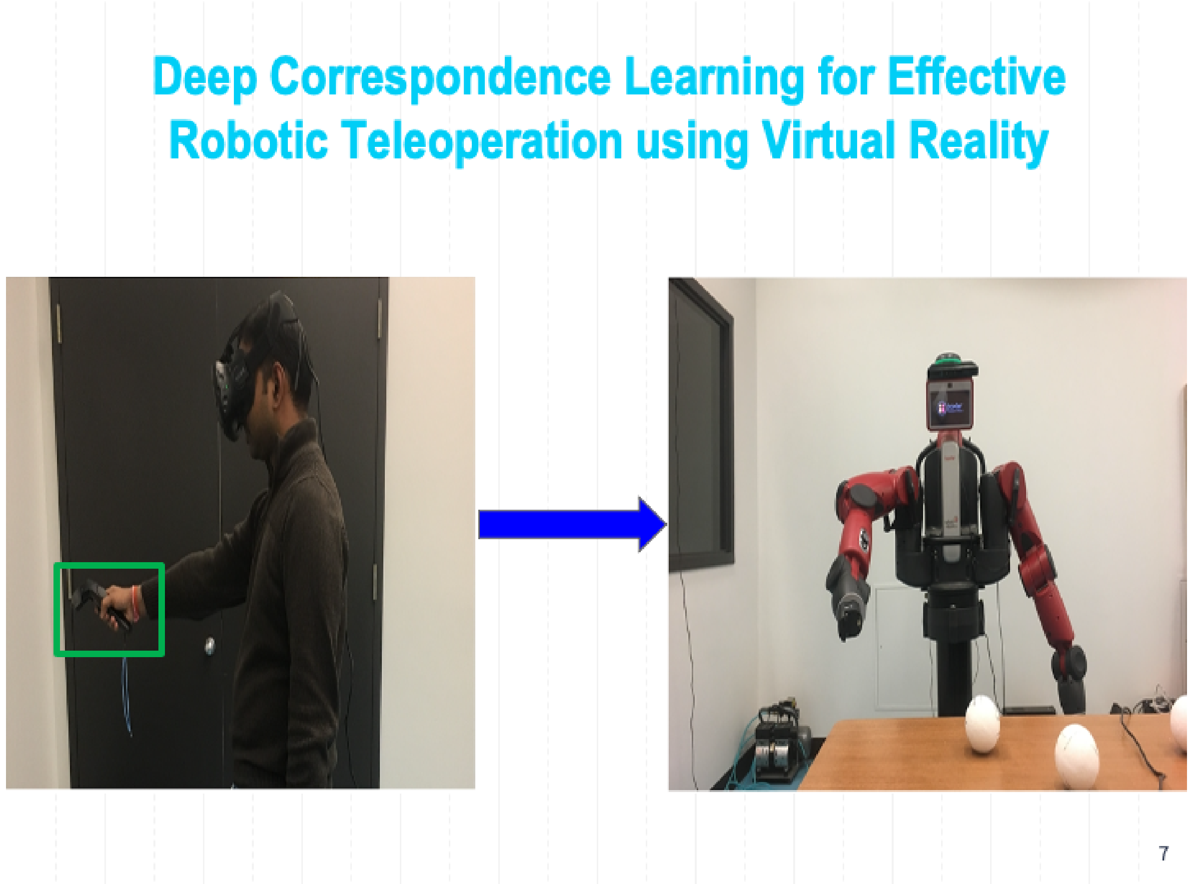


Figure 1. How can we effectively teleoperate a robot in virtual reality using deep correspondence learning?

dimensional robot joint angles is a non-linear transformation problem as shown in Figure 1. This chapter proposes a deep correspondence learning model (feed-forward neural network) to solve this non-linear transformation of HTC Vive positions to robot joint angles.

One of the problems in a robotic teleoperation system is predicting human (the teleoperator) intention for the robot to be prepared for conducting appropriate actions. In general, for a robot

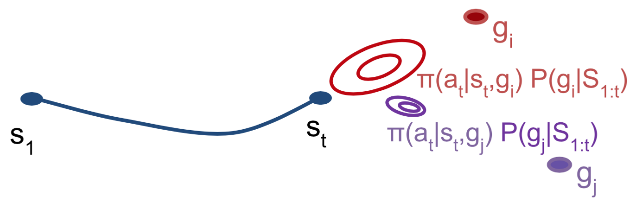


Figure 2. Can we predict human intention more quickly based on the partial trajectory traveled by maximizing the goal likelihood for Inverse Optimal Control?

to be an effective peer in a typical human-robot collaborative setting it needs to predict human intention to better plan its actions. For example, in a typical self-driving car scenario, it is important to predict pedestrian's intentions and other driver's intentions to take suitable actions on the road. Motivated by these problems, the second chapter address the problem of human intention prediction. The goal of this problem set is for the robot to not only predict human intentions correctly, but also more quickly. It explores maximum entropy inverse reinforcement learning (MaxEnt IRL) algorithms for goal predictions based on partial trajectory traveled by the human. In a typical MaxEnt IRL setting, the likelihood of the trajectory is maximized for training the reward parameters (21; 22). However, at inference time the aim is to predict the true goal given the partial trajectory traveled i.e., maximizing the goal likelihood as shown in Figure 2. This second chapter attempts to fill the research gap by proposing an algorithm for discriminatively learning MaxEnt IRL for true goal predictions based on partial trajectory traveled. The proposed algorithm is evaluated on a goal pointing dataset (3) and Cornell Activity Dataset (CAD-120)(23).

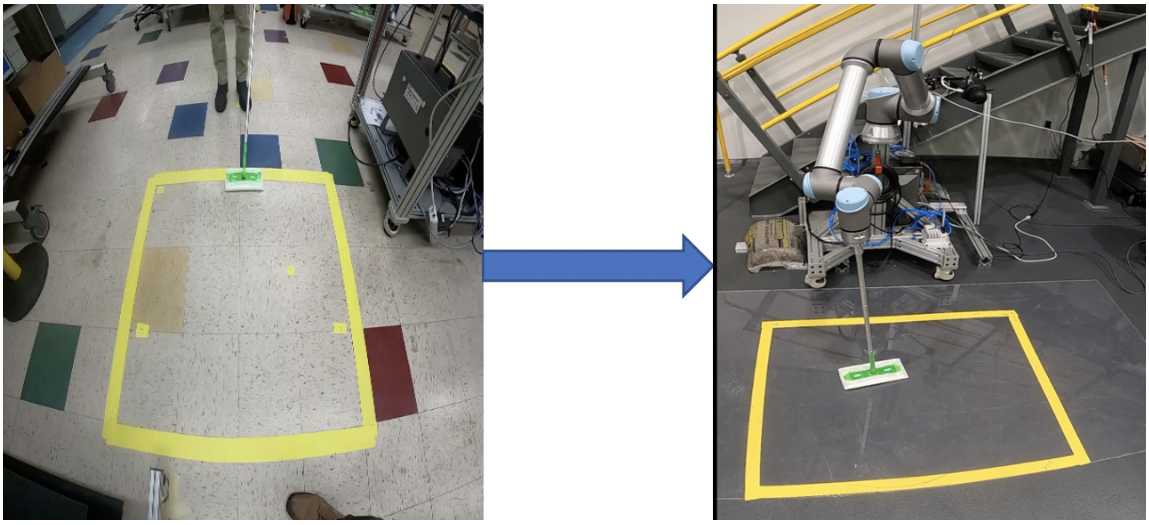


Figure 3. Can a robot learn a day-to-day task (for example, mopping the floor) from a human demonstration provided in the form of a video?

Mopping the floor is one daily task that is not very exciting for humans, so it would be great if a robot could perform this action instead. The final work of the thesis explores robot learning for this tedious daily task. One potential approach is to program a robot to mop by mimicking human mopping behavior. However, human motion behavior is very difficult to capture with hand-coded robotic programs. Another potential approach is using imitation learning, which is robot learning from human demonstration. A human demonstration can be recorded using motion capturing sensors that are stuck to the human body to accurately measure human motion. However, this could bring some discomfort to the human demonstrator. One final consideration is the recent advancement in the field of computer vision (4) which helps a robot imitate a video of recorded human behavior from a mobile camera on a robotic arm.

The final chapter of the thesis explores imitation learning using video demonstrations of humans performing mopping tasks. First and third-person videos of humans mopping the floor using a floor mop are collected. Two robotic systems are build to learn mopping motions on a robotic arm from human video demonstrations. The first robotic system uses a traditional computer vision technique to track the mop in human videos and then uses inverse kinematics to move the robotic arm attached with a mop. The second system comprises advanced computer vision techniques on Time Contrastive Network (TCN) and Reinforcement Learning (RL). Using the collected videos, a TCN model is learned. The TCN produces embeddings that map similarities in images from two different viewpoints that are visually different but temporally the same. Also, the TCN can differentiate between images from the same viewpoint that may look similar but are temporally different. A mopping task is learned on a UR 10e robotic arm using reinforcement learning (RL). The reward function is devised for the complex mopping task comprising of TCN embedding of human demonstration mopping video versus robot first-person mopping video feed and z-axis value of the mop. After 15 iterations of RL, a robotic arm can mimic human mopping behavior following video demonstration. The third-person learned robot mopping behavior video is compared with the human third-person video demonstration using cosine similarity of the optical flow of the mop head in both the videos at each frame.

1.1 Outline of the document

This Ph.D. thesis document is organized into five parts: Introduction; Deep Correspondence Learning for Effective Robotic Teleoperation Using Virtual Reality; Discriminatively Learning

Inverse Optimal Control Models for Predicting Human Intentions; Robot Learning to Mop like Humans using Video Demonstrations; Conclusion and Future Work.

The first chapter introduces the problems in teleoperation, human intent prediction and imitation learning methods for collaborative robot space. Then it outlines the proposed solution to three problems addressed.

Chapter two is derived from research paper titled Deep Correspondence Learning for Effective Robotic Teleoperation using Virtual Reality (24) published at IEEE International Conference on Humanoid Robots (ICHR) 2019. This chapter introduces the correspondence learning problem for robotic teleoperation using Virtual Reality and discusses the background and related works. Further, deep correspondence architecture is explained as an approach to solving this problem. Lastly, this chapter explains the conducted experiments and results obtained.

Chapter three is obtained from the research paper titled Discriminatively Learning Inverse Optimal Control Models for Predicting Human Intentions (1) published at ACM International Conference on Autonomous Agent and Multi-Agent Systems (AAMAS) 2019. It explains the motivation behind the human prediction task prior work in that area. Then the algorithm is proposed for learning Inverse Optimal Control models for maximizing goal likelihood. Further, the proposed hypothesis is confirmed using two data sets and the results are discussed.

The work presented in Chapter four is submitted to Conference on Robot Learning (CoRL), 2021 for review. This chapter aims to answer the question: Can a robot learn a day-to-day task (for example, mopping the floor) from a human demonstration provided in the form of a video? The chapter introduces the problem of learning a daily task from video demonstration and then explains our two proposed robotic system to mop the floor. Next we present experiments conducted, results obtained and other related work to imitation learning for mopping task.

Lastly, chapter five gives the concluding statement for this thesis and directs to the future extensions based on this thesis work.

CHAPTER 2

DEEP CORRESPONDENCE LEARNING FOR EFFECTIVE ROBOTIC TELEOPERATION USING VIRTUAL REALITY

(Previously published as S. Gaurav, Z. Al-Qurashi, A. Barapatre, G. Maratos, T. Sarma and B. D. Ziebart, "Deep Correspondence Learning for Effective Robotic Teleoperation using Virtual Reality," 2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids), 2019, pp. 477-483, doi: 10.1109/Humanoids43949.2019.9035031.)

2.1 Introduction

Tasks like performing experiments with hazardous substances or handling dangerous waste (6; 10) are most safely performed by humanoid robots. However, often these tasks cannot be completed successfully without human guidance. Teleoperation is one of the most popular ways used for a human operator to remotely guide and control humanoid robots (5; 6; 25; 26; 7; 8; 9; 10). Traditionally, a teleoperator remotely watches the robot's environment projected on a screen (2-D view) from cameras mounted on the robot's head (26; 25) and uses a joystick (7; 8) or controller (9; 27; 28; 6) to operate the robot.

Depth cameras (e.g., the Microsoft Kinect) have been employed as the input sensor to control a Baxter robot in previous work (9; 17; 18; 3; 19; 20). However, depth cameras often suffer from sensor noise that can produce errors and poor translation for robotic teleoperation when mapping from a tracked teleoperator skeleton to robotic joint positions. Additionally,

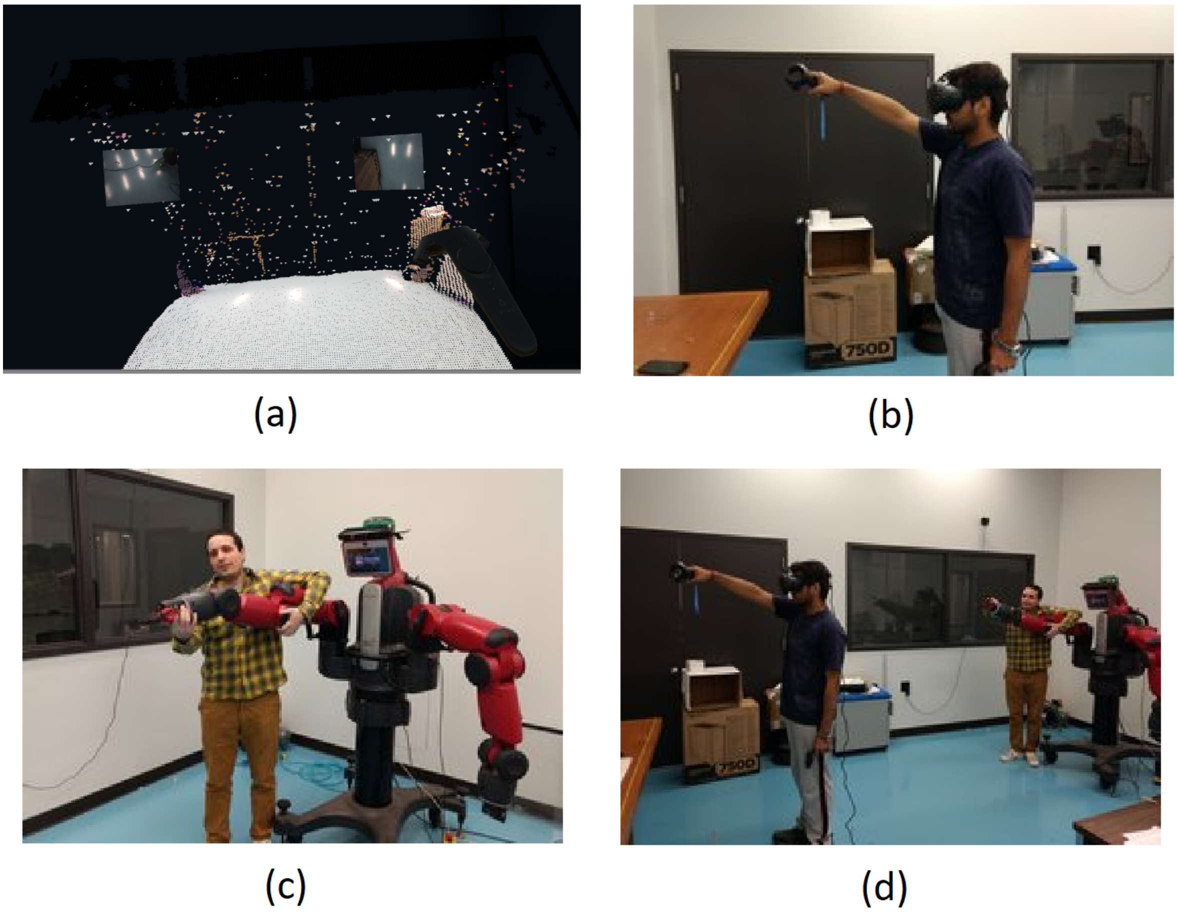


Figure 4. (a) Visualization of the Baxter robot's workspace captured from a depth camera (Microsoft Kinect) and displayed on a virtual reality headset (HTC Vive); (b) demonstrator holding HTC Vive controller slowly moves his hand from neutral position; (c) trainer moves robot hand manually to follow human trajectory; and (d) demonstrator and trainer with Baxter reaching the same configuration.

having only the 2-D view of the robot's workspace often makes it difficult for the operator to precisely control the robot within its environment.

In contrast, recent advances in virtual reality technologies provide an opportunity to address this difficulty. Virtual reality is popular for creating virtual environments of any room or workspace (14; 13). Virtual reality systems like the HTC Vive typically incorporate two controllers held by an operator that can track the human wrist with very high accuracy and allow the operator to realistically manipulate objects in the virtual environment. Recently, robotic teleoperation has been performed using 3D sensors like the HTC Vive and Oculus Rift (11; 12; 13; 14; 15; 16).

The teleoperation correspondence problem of mapping from teleoperator poses or controls to robot poses is a crucial problem for enabling robotic teleoperation using a virtual reality system. Existing methods use linear coordinate transfer from a virtual reality frame of reference to a robot's frame of reference and then perform inverse kinematics to move a robot's arm (11; 12; 13; 14; 15). This translation mechanism can be slow and erroneous due to multiple joint configuration solutions being provided by inverse kinematics for a single point.

In this chapter, we propose a machine learning approach for estimating an appropriate non-linear correspondence for robotic teleoperation from human pose. We consider teleoperating the Baxter robot using a Microsoft Kinect depth camera for perceiving the robot's workspace and an HTC Vive virtual reality system for visualizing the workspace and providing 3-D control, as shown in Figure 4. First, we collect correspondence positions of human end-effectors and Baxter joint angles by asking a demonstrator holding an HTC Vive controller to move his or her hand while an operator moves a Baxter arm in synchronization with the human demonstrator (Figure 4). Second, we explore different non-linear machine learning regression models as baseline correspondence models. Next, we explore deep learning architectures to learn a non-linear correspondence model for HTC Vive controllers to Baxter Robot joints. We show that our proposed deep correspondence model performs significantly better than linear and non-linear regression baselines and helps to enable more effective robotic teleoperation using virtual reality. To demonstrate the effectiveness of our proposed model, we conduct a simple real-life experiment: picking up a box from the table and placing it at another location. Our proposed deep network enables the teleoperator to perform the task faster and more effectively than the baseline methods.

The chapter is organized as follows: we first provide related work on robotic teleoperation using virtual reality. Then, we describe in detail our deep correspondence architecture for robotic teleoperation. Next, we describe the experiments we conducted to compare our correspondence learning approach with baseline methods and discuss the results. Lastly, we conclude the chapter and propose future work.

2.2 Background and Related Work

2.2.1 Correspondence Learning in Virtual Reality

Virtual reality (VR) can provide a teleoperator with a first-person perspective from a robot's viewpoint (13). This enables high-quality demonstrations for robotic manipulation to be collected (29). Fritsche et al. (13) use the Oculus Rift and Microsoft Kinect camera as teleoperation input and iCub as their humanoid robot. Their correspondence (transfer of human embodiment to the robot embodiment) is accomplished via Kinect camera skeleton tracking. The virtual reality setup is only used to give the first-person perspective for performing the task. As mentioned in the previous section, the Kinect camera can produce erroneous translations, preventing effective teleoperation. In this chapter, we directly learn controls from human end-effector to robot embodiment via a deep learning approach.

Previous work uses consumer-grade virtual reality headsets (HTC Vive) supported by hand tracking hardware that can be used to naturally teleoperate robots to perform complex tasks with some delay (11; 14; 15). Zhang et al. (11) teach the robot via demonstrations collected in virtual reality.

These prior works (11; 14; 15) use linear coordinate transfer from virtual reality frame of reference to the robot frame of reference for correspondence learning. Then, they use inverse kinematics to find robot joint angles to move the arm. Unfortunately, inverse kinematics can provide different joint configurations for one end-effector of a robot arm. This may lead to an undesirable joint setting that produces irregular arm movements and ultimately sometimes fails to accomplish a task. Also, the whole process is complicated and time-consuming. We use a linear regression baseline method to emulate these prior work for comparison.

2.2.2 Baxter Robot

The Baxter robot is built by Rethink Robotics. It has a torso mounted on a movable pedestal and two arms on the left and right sides of the robot (30). Each arm has seven degrees of freedom (DOF), i.e., seven joint angles:

$$R_{\text{joints}} = [s_0, s_1, e_0, e_1, w_0, w_1, s_2]. \tag{2.1}$$

Forward kinematics[1] provides the end-effector:

$$R_{\text{end-effector}} = [x_t, y_t, z_t, x_r, y_r, z_r, w_r], \tag{2.2}$$

where the first three are translation points describing the position of Baxter's arm end-effector and the last four are Quaternion angles describing the rotational position.

2.2.3 Deep Learning Architecture

Deep neural networks been developed to address prediction tasks for which more conventional computation approaches have proven ineffective (31; 32). They are attractive for computing the inverse kinematics and dynamics of robots because they can be trained for this purpose

---

[1]http://sdk.rethinkrobotics.com/wiki/Baxter_PyKDL#baxter_kinematics.py

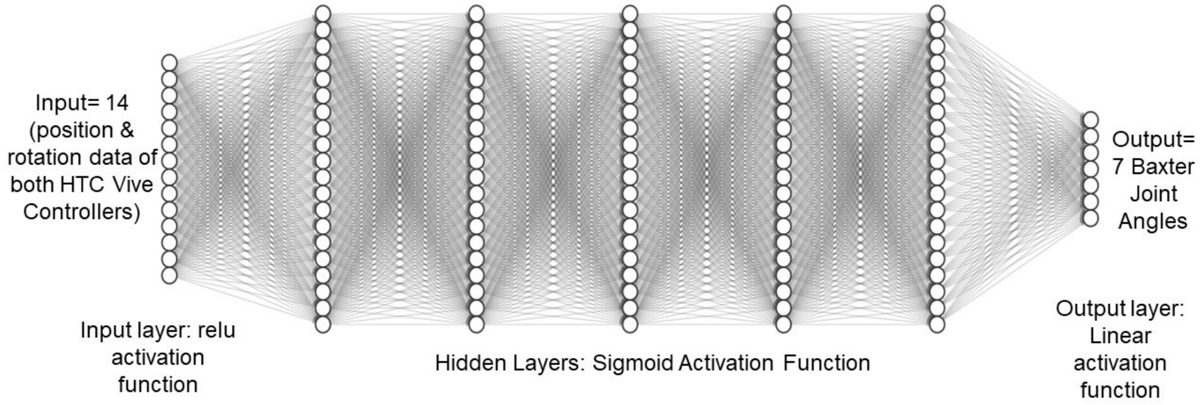


Figure 5. Training of the deep neural network to learn the correspondence between the Baxter robot joint angles and the human poses where input 14 represents the position and rotation data of HTC Vive controller placed in two respective hand of the human demonstrator and output 7 represents the seven joint angles of one of the Baxter robot's arm.

without explicit programming (33) and can represent complex non-linear relationships. The basic operation carried out at a single neuron is represented as:

$$a_i = f^i(w_i a_{i-1} + b_i), \tag{2.3}$$

where a_i is the output of layer i, $f^i()$ is the activation function of layer i, w_i is the weight matrix between layer i and layer i-1 and b_i is the bias of layer i. Several algorithms have been developed to train a neural network, including back propagation (with momentum) and Levenberg Marquaidt algorithms (34) (35). We use deep networks for our correspondence model in this chapter.

2.2.4 Loss Function

The loss or evaluation functions used to evaluate our correspondence learning models are:

2.2.4.1 Mean Squared Error

The Mean Squared Error (MSE) is calculated by computing the squared difference between actual value (Y) and predicted value ($\hat{Y}$) and averaging over total number of values,

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2. \tag{2.4}$$

We employ this measure to train our deep network model and evaluate the performance of its resulting predictions.

2.2.4.2 Cosine Similarity

Cosine similarity measures the similarity between two non-zero vectors of an inner product space based on the cosine of the angle between them:

$$\cos(\theta) = \frac{\mathbf{AB}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}, \tag{2.5}$$

where A and B are vectors. We use cosine similarity to compute the loss of rotation angles of the end-effector.

2.3 Approach

In this section, we first describe the visualization of the Microsoft Kinect depth data in our virtual reality system. Then, we explain the underlying problem of non-linearity in the correspondence from the HTC Vive controller to the Baxter joint angles. Lastly, we propose in detail our main contribution: a deep correspondence learning architecture.

2.3.1 Visualizing Microsoft Kinect depth data in Virtual Reality

We first develop a visualization of the Kinect depth point cloud data in the HTC Vive. In the Unity application, we read the sensor data from a Kinect V1, which has a resolution of 640 x 480 pixels and an 11-bit depth. We use this collected sensor data to render a mesh in Unity in real-time to create a point cloud representation. The color information from the collected data is also maintained in the representation as it makes objects more discernible. We scale and orient this point cloud to elicit realism in the visualization using manually selected parameters. Using the Unity SteamVR plugin, we integrated the HTC Vive in the application to view the point cloud in virtual reality. Figure 4(a) shows the visualization of Baxter's environment in HTC Vive headset.

The 3-D projection of the robot's environment provides teleoperators with a first-person perspective for performing tasks effectively. However, to accomplish a precise grasping task, we stream frames from the Baxter wrist camera (2-D) to the HTC Vive headset. Using the Unity engine, we display the two wrist cameras by streaming them on the top corners, as shown in Figure 4(a). This can help the teleoperator perform a fine-grained task when the robot arm has reached close to the target location.

2.3.2 Robotic Teleoperation using Virtual Reality

The crux of any robotic teleoperation system relies on the transformation of the human operator's control inputs to the robot joint values. For a virtual reality system, this involves a transfer of embodiment from the human to the robot that is called a correspondence. We consider the setting in which the human operator holds an HTC Vive controller in each hand and wears the HTC Vive headset to view the Baxter robot's 3D environment from the first-person perspective. The HTC Vive data consists of 21 data points, which comprise positions (translations x_t, y_t, z_t) and orientations (quaternion angles x_r, y_r, z_r, w_r) of the HTC Vive headset and two HTC Vive controllers. The orientation is provided to us in quaternion by the HTC Vive system because it has computational advantages. We understand orientation is represented as three degrees of freedom. However, we did not convert the quaternion to Euler angles because it won't have any effect on the overall training of the deep models (36; 37).

The two HTC Vive Controller positions indicate the two human end-effectors. Mapping each position to the seven joint angles of each Baxter arm is challenging due to non-linearity. To verify this, we compared a linear model with polynomial models to regress each one of the joint angles from the HTC Vive data. The polynomial models outperform the linear models with the result being statistical significant with a P-value $< 10^{-16}$, thus proving that it is indeed a non linear transformation.

2.3.3 Deep Correspondence Learning Architecture

Deep learning has proven to be an effective method for non-linear function approximation, as explained in Section 2.2.3. In this chapter, we propose a fully connected deep neural network architecture for our correspondence model, as shown in Figure 5. The data points from two HTC Vive Controllers are used as input to the deep network, and the output is the required seven joint angles of the Baxter Robot arm. We choose the position of both hands (14 data points) versus only the controlling hand (seven data points) as input because the location of the other non-controlling hand helps the model better situate the relative position of the controlling hand in the space. We use off-line training for the deep network as shown in Figure 5. The deep network consists of five hidden layers (reason explained in Section 2.4.5) each with 64 neurons. The motivation for deciding neurons in the hidden layer comes from the statistical difference between human end-effector position to the Baxter joint angles as explained in the previous section.

We use a Rectified Linear Unit (ReLU) activation function (also known as the ramp function) $f(x) = x^+ = \max(0, x)$, in the input layer, a Sigmoid response ("S"-shaped curve) $f(x) = \frac{1}{1+e^{-x}}$, in the five hidden layers, and a linear activation function $f(x) = cx$, in the output layer in order to obtain an output in radians between -1 to 1.

We use the mean squared error (MSE), as the loss function, $\frac{1}{n}\sum_{i=1}^{n}\left(\theta_{ip} - \theta_{id}\right)^2$, where θ_{ip} is the predicted output joint angles and θ_{id} is the desired joint angles. We use ADAM (38) as the optimizer to converge quickly to small loss.

2.4 Experiment

2.4.1 Hardware Setup

We evaluate our approach using a Baxter robot as our experimental robot in this chapter. We use a Microsoft Kinect Depth V1 camera with frequency up-to 20Hz mounted to the head of the Baxter robot to visualize the robot work-space or environment, and an HTC Vive as our virtual reality platform, which publishes data with a frequency of up to 90Hz. The HTC Vive comprises one headset, two controllers and two base stations for tracking. We use a Windows machines for running the VR platform that has an i7 processor, 16 GB of RAM and a GTX 1080 ROG graphics card. We use ROS (Robot Operating System) to communicate with the hardware. We also use the Baxter wrist camera to stream frames to the HTC Vive headset with a frequency of up to 25Hz.

2.4.2 Data Collection

A demonstrator is asked to wear the HTC Vive headset where he/she can view what Baxter sees as shown in Figure 4-a. Also, the same demonstrator is requested to hold the two HTC Vive controllers in respective hands. This demonstrator then moves one of their hands slowly for 3-5 minutes in random directions. The trainer moves the corresponding arm of Baxter in synchronization to the demonstrator's movement, as shown in Figure 4(c). Both the demonstrator and Baxter's arm start moving from the same starting position in space. This data collection process is conducted for both arms separately. The attached video contains more details of the training data collection.

There are three components of the HTC Vive used in this setup, thus 21 data points (7 points for each component) are transferred over the network to the Robot Operating System (ROS) environment. These 21 data points from the HTC Vive and the 7 joint angles of one Baxter arm are recorded at 40 Hz frequency. The collected data are 44,941 data points for the right hand and 36,155 data points for the left hand from 11 demonstrators.

2.4.3 Data Pre-processing

The 11 demonstrators who contributed to data collection are of different height and arm length. To generalize our training, we subtract the HTC Vive headset translation positions

(x_t, y_t, z_t) from both HTC Vive controllers' translation positions to measure the controllers' movements relative to head positions. We randomly withheld one person's trajectory data for conducting an offline experiment. On the remaining ten demonstrators' data, we apply leave-one-out-cross-validation (LOOCV), i.e., training on nine people's data and testing on the tenth. We repeat the process with each person's data and take the average.

2.4.4 Training of Baselines

We compare our deep correspondence model against a linear regression approach (3) with either no, polynomial, or Gaussian expansion of the feature space, and support vector regression with polynomial or radial basis function (RBF) kernels. The linear regression method is used as a state-of-the-art approach to the correspondence learning problem. We observe that the difference in average Euclidean loss between linear regression with and without expansion of the feature space is significant, which is further evidence that the correspondence task is non-linear in nature. The detailed result of each baseline is reported in Table I.

2.4.5 Training of Deep Learning Models

Motivated by the deficiencies of the linear and non-linear baseline models, we investigate deep learning models. We started with a simple network having one hidden layer and experimented with different activation functions (e.g., hyperbolic tangent, Sigmoid, rectified linear) for input and output layers. For each configuration, we trained until reasonable validation loss was obtained. We then tested the performance of the correspondence model on the Baxter robot. The configuration of rectified linear activation functions for the input, Sigmoid activation functions for the hidden layers, and linear activation functions for the output layer gave the best correspondence. We then tested different numbers of layers (up to 20 layers) and different numbers of neurons (e.g., in multiples of 16) in each layer. Using the best of numbers of layers and number of neurons, we then trained our deep model to convergence with a difference of training losses for the last ten epochs of less than 0.0001.

2.4.6 Experimental Setup

We conducted two types of experiments to evaluate our proposed model against the linear and non-linear baselines.

2.4.6.1 Offline Experiment

We first find the difference between the predicted output and the ground truth for the baselines and our deep model. We randomly selected a demonstrator's trajectory from the Vive-Baxter correspondence dataset comprising of 2500 pairs of poses. The corresponding Baxter's joint angles of this trajectory (actual/desired value) are considered as ground truth. The trajectory is passed to the linear regression correspondence model, the best non-linear regression model, and the deep network correspondence model. The respective output joint angles from both models are recorded. We apply forward kinematics to output Baxter's arm joint angles (ground truth, linear regression, non-linear regression, and deep network) to calculate Baxter's arm end-effectors, as shown in Figure 7. The simultaneous movement of the Baxter robot arm using position control for each of the four models is demonstrated in an attached video.

We compute the Euclidean distance of Baxter's end-effector as provided by the correspondence model with the ground truth position, as shown in Table II. To measure the loss in rotational angle of Baxter's end-effector, we compute the cosine similarity between the pre-

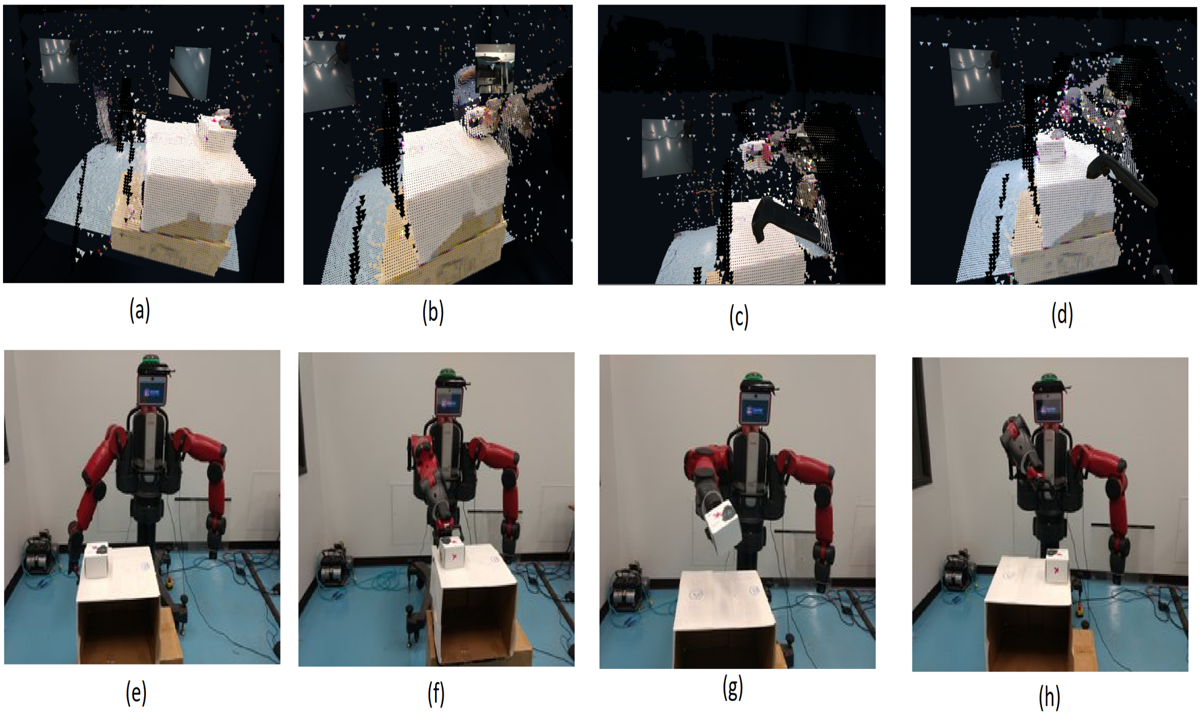


Figure 6. (a) Pick and place experiment setup in virtual reality, (b) Teleoperator reaching the object (white box) kept at position A in virtual reality, (c) Teleoperator picking up the white box from position A and moving to position B in virtual reality, (d) Baxter teleoperated to reach the object (white box) kept at position A, (e) Pick and place experiment setup in reality, (f) Baxter teleoperated to reach the object (white box) kept at position A, (g) Baxter teleoperated to pick up the white box from position A and moving to position B, (h) Baxter teleoperated to place the box at position B.

dicted angles and the ground truth as shown in Table III. The graph between predicted end-effector from linear regression vs deep network vs ground truth is plotted in Figure 7 for three translation position and four rotational angles respectively.[1]

2.4.6.2 Real-time Robotic Teleoperation

We use a simple pick and place task (Figure 6) for our second set of experiments. A teleoperator is asked to wear the HTC Vive headset where he/she can view the virtual robot environment, as shown in Figure 6(a). The teleoperator must move the Baxter arm from a neutral position to position A where a white box is placed, grasp the box, and move it from position A to position B within one minute to be considered successful. Figure 6(a)-(d) shows the steps conducted by the teloperator in the Virtual reality and Figure 6(e)-(h) demonstrates corresponding steps performed by Baxter robot using the proposed deep correspondence model. This task is repeated by two different teleoperators using linear regression, the best performing non-linear model (SVR-RBF), and our proposed deep architecture. There are five trials for each model and each of them is randomized. Thus, we collected ten samples for each model to compare success rate.

In real-time, the two HTC Vive controller positions (after subtracting the HTC Vive headset's translation position) are passed to the trained model. The output (7 Baxter's arm joint angles) from the trained correspondence model is then passed to the Baxter, which moves its arm using position control, as shown in Figure 6. The average time taken to complete this task

---

[1] SVR (RBF) does not provide significant improvement, but clutters the differences between deep and linear, so we do not include it.

TABLE I

RESULTS (TEST LOSS IN RMSE) OF HTC VIVE CONTROLLER TO BAXTER CORRESPONDENCE MODELS

Model	**Left Arm (rad)**	**Right Arm (rad)**
Linear Regression	0.6430	0.6699
Kernel Regression (Poly=2)	0.5808	0.5840
Kernel Regression (RBF)	0.5788	0.6196
SVR (Poly=2)	0.6019	0.4863
SVR (Poly=3)	0.5716	0.4758
SVR (RBF)	0.4944	0.4567
Deep Networks	0.0735	0.0659

TABLE II

EUCLIDEAN DISTANCE MEASURE FROM GROUND TRUTH BAXTER'S END-EFFECTOR POSITION TO PREDICTED BAXTER'S END-EFFECTOR

Model	**Euclidean Loss (m)**
Linear Regression	0.3414
SVR (RBF)	0.2173
Deep Network	0.0267

and the success rate of each model is reported in Table IV. The detailed video on collection of training data and the pick and place experiment can be found here.

2.5 Results and Discussion

Table I describes the evaluation results of the HTC Vive controller to Baxter arm joint angle correspondence. As shown in this table, the MSE for the Baxter joint angles using the simple linear regression model is very high (i.e., a poor correspondence). All of the four non-

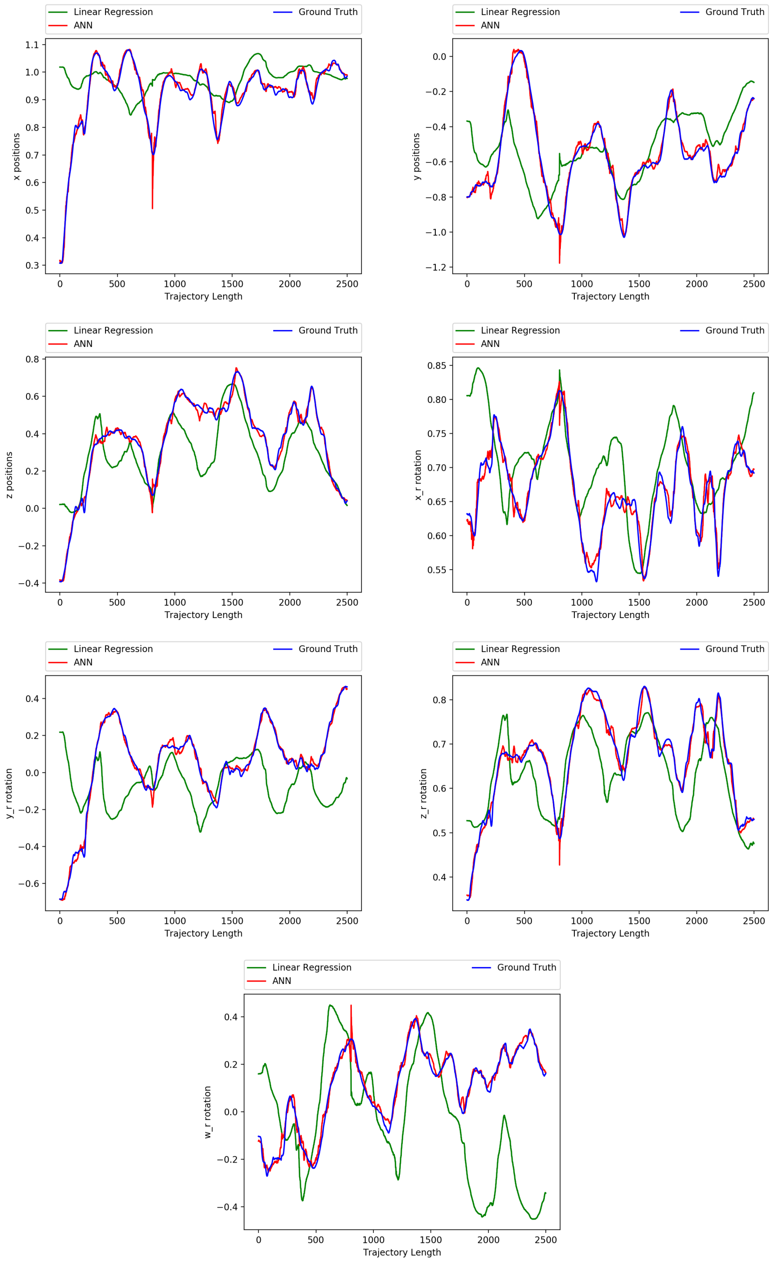


Figure 7. (a)-(c) Robot end-effector positions (x_t, y_t, z_t) plotted for Ground Truth vs Linear Regression vs Deep Network; (d)-(g) Robot end-effector rotational angles (x_r, y_r, z_r, w_r) plotted for Ground Truth vs Linear Regression vs Deep Network.

TABLE III

COSINE SIMILARITY RESULT

	Linear Regression	**SVR (RBF)**	**Deep Network**
x_r	0.467	0.628	0.986
y_r	0.863	0.876	0.976
z_r	0.560	0.785	0.985
w_r	0.160	0.428	0.959

TABLE IV

SUCCESS RATE OF PICK & PLACE EXPERIMENT

	Success Rate (%)	Average Time (sec)
Linear Regression	60	56
SVR (RBF)	70	50
Deep Network	80	41

linear models performed better than the linear regression model and Support Vector Regression (SVR) with a Radial Basis Function (RBF) kernel performed best on the LOOCV test. Thus, we selected SVR (RBF) as our non-linear baseline model.

The deep network provided substantially lower error, with a decrease in MSE for the left hand by a factor of 8.74 compared to the linear model and a factor of 6.73 to SVR (RBF). A similar decrease in MSE can be seen for the right arm. This demonstrates that our proposed deep architecture more successfully models the non-linearity in the Vive-Baxter correspondence data.

Table II shows the Euclidean distance between the robot's ground truth end-effector and predicted end-effector positions. According to Table II, the Euclidean distance loss between linear regression to ground truth robot's end-effector is 0.3414 meters and SVR (RBF) non-linear model to ground truth is 0.2173 meters. On the other hand, the Euclidean distance loss between ground truth using the deep network is only 0.0267 meters. Also, the attached video clearly demonstrates the difference of performance across the four models (ground truth, linear regression, non-linear regression, and deep network) in the real teleoperated arm movement of the robot. Thus, we have demonstrated significant amounts of improvement in predicted end-effector by our proposed deep network compared to linear and non-linear baselines.

Table III shows the cosine similarity between the rotational angles of the predicted end-effector with the ground truth. For all four rotational angles, our deep model outperforms the baseline linear and non-linear regression by a large margin. Therefore, our deep correspondence model enables more appropriate control of end effector orientation for fine-grained manipulation tasks.

The translation positions (x_t, y_t, z_t) and rotational angles (x_r, y_r, z_r, w_r) are plotted for Baxter's end-effector for ground truth, linear regression and our deep network in Figure 7. We infer from these seven graphs that the deep learning model follows the ground truth very closely whereas linear regression is far away from the ground truth.

Figure 6 shows snapshots of Baxter performing a pick and place task using the deep correspondence model. We report the results of this experiment in Table IV. Success rates and average completions times both improve from the linear regression model to the non-linear regression model (SVR with RBF kernel), and from the non-linear model to our deep learning approach. We find that our approach provides the highest success rate and lowest average completion time. Thus, our offline performance successfully transfers to real-time control.

2.6 Conclusion and Future Work

We have successfully trained correspondence models for HTC Vive Controller to Baxter's arms. Our proposed deep model achieves better results than linear and non-linear regression baseline models for correspondence-based evaluations. In the real-time experiment, our deep network performed better than baselines model, resulting in faster completed tasks.

In this chapter, we have investigated the problem of correspondence learning for teleoperating the hands of a humanoid robot. In future work, we plan to extend our approach to a complete humanoid robot (e.g., Nao) (13; 12; 5; 9; 6; 28). This will help better collect training data for teaching humanoid robots and also perform effective real-time humanoid robotic teleoperation.

We are planning to evaluate more complex tasks, like "opening a jar" and tasks mentioned by Whitney et al. (2018) (15). To improve the completion time and smooth teleoperation, we are planning to apply methods for goal predictions (3; 1) using our trained HTC Vive to Baxter arm correspondence model. This will help the teleoperated robot reach the goal in less time and with less distance traveled compared to previous work (3).

In the future, we would like to improve the visualization of the robot's environment in virtual reality using better 3-D cameras and better visualization techniques. We would like to incorporate better data collection techniques involving HTC Vive (29). Moreover, we are planning to apply other deep learning techniques such as recurrent neural networks that accommodate the sequential nature of points in the trajectory.

2.7 Acknowledgment

We thank the demonstrators and trainers who volunteered for data collection. This research is supported as part of the Future of Life Institute (futureoflife.org) FLI-RFP-AI1 program, grant #2016-158710 and NSF grant #1652530.

CHAPTER 3

DISCRIMINATIVELY LEARNING INVERSE OPTIMAL CONTROL MODELS FOR PREDICTING HUMAN INTENTIONS

(Previously published as S. Gaurav and B. Ziebart, "Discriminatively Learning Inverse Optimal Control Models for Predicting Human Intentions," 2019 In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '19). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, pp 1368–1376.)

3.1 Introduction

Humans and robots work in close collaboration for many tasks (39; 40; 41; 42; 43; 44) or simultaneously pursue separate tasks in shared workspaces (45; 46; 47; 48). To enable effective task completion in either setting, robots should be able to anticipate human intentions prior to the completion of the pursued task. Doing so enables a robot to plan compatible actions ahead of time that are more productive for collaborative tasks or with fewer resource conflicts in separate tasks. For example, self-driving vehicles that can predict pedestrians' intentions and behaviors can navigate more safely and efficiently at intersections. However, improved methods for predicting human intentions are needed to support these examples of more synergistic decision making in autonomous systems.

Bayesian reasoning has been predominantly used to address the goal prediction task. Under this perspective, a predictive model of the trajectory of decisions given the goal is employed—along with a prior distribution over goals—to obtain the posterior distribution over goals. Numerous methods for the trajectory likelihood model have been employed (49; 23; 22; 50; 51; 52), ranging from simple goal-conditioned Markov models (53; 54) to inverse planning (52) and imitation learning methods (50). Central to all of these methods is that the trajectory likelihood models are designed and optimized with sole consideration to trajectory prediction rather than goal prediction. While Bayes theorem holds for the true distributions of goal posteriors and trajectory likelihoods, it can produce error-prone goal posteriors when the likelihood model is noisily estimated from limited amounts of available data.

In this chapter, we investigate training maximum entropy inverse reinforcement learning models (21) to maximize goal prediction likelihoods rather than trajectory likelihoods. In section 3.3, we develop our method for calculating the gradient for the likelihood of the final goal. By experimenting with an object reaching task with trained reward function from our new approach, we realize an average probability for the true goal given approximately 50% of the trajectory traveled that is not realized until 70% of the trajectory is traveled using the trajectory-based likelihood method (22). Also, we also evaluate our method on the Cornell CAD-120 dataset (23).

The chapter is organized as follows: we start with a summary of background information on decision processes, previous work on predicting human intention, the inverse optimal control formulation for imitation learning, and goal prediction using an inverse linear-quadratic regulation (LQR) formulation. Next, we describe in detail our algorithm for obtaining goal predictions from the MaxEnt IRL model trained using goal likelihood maximization rather than trajectory likelihood maximization. Next, we explain the experimental setup used to evaluate our proposed method. The result section summarizes the results obtained by our goal likelihood method versus the trajectory likelihood method and other baselines. Lastly, we provide conclusion and propose future work.

3.2 Background and Related Work

3.2.1 Decision Processes and Goal Prediction

A wide variety of tasks can be represented using sequential decision process formulations. A Markov Decision Process (MDP) is defined[1] as a tuple $(\mathcal{S}, \mathcal{A}, \tau,$ R), where:

- state S is from a finite set of states s $\in \mathcal{S}$;
- action A is from a finite set of actions a $\in \mathcal{A}$;
- τ is the state transition probability from state s under action a;
- R(s_t) is the reward or cost received by visiting state s_t.

A sequence of states and actions, $s_1, a_1, s_2, a_2, s_3, \ldots, s_T, a_T$, is produced by applying a decision policy $\pi(a_t|s_t)$ to the state transition dynamics of the decision process, $\tau(s_{t+1}|s_t, a_t)$.

In many domains, the decision processes for similar tasks differ only in small ways. We consider these differences being parameterized by a goal state g (where $g \in \mathcal{G}$, is set of all

---

[1]We denote random variables with uppercase letters, fixed variables in lowercase, and matrices in boldface uppercase.

possible goals in the environment) that indicates the successful accomplishment of the goal when it is reached at final time step t_f (i.e., $s_{t_f} = g$). In contrast, if the goal state is not reached (i.e., $s_{t_f} \neq g$), a large cost (or negative reward) is incurred. More generally, the reward function can be parameterized by the goal g as: $R_g(s)$.

In this chapter, we also consider continuous-valued states and actions that can be modeled using a linear-quadratic regulation (LQR) formulation. In LQR, the dynamics of a system being investigated are represented by a linear relationship,

$$s_{t+1} = \mathbf{A}s_t + \mathbf{B}a_t + \epsilon_t, \tag{3.1}$$

where s_t denotes the state of the system at time t, a_t denotes the action at time t, ϵ_t denotes some zero mean Gaussian noise, and $\mathbf{A}$ and $\mathbf{B}$ define the system dynamics. The Equation 3.1 contains position and velocity term which corresponds to the dynamics by which robot is driven i,e. the torque and force applied to move motors which in turn causes motion in robot. The state-action cost function,

$$cost(s_t, a_t) = \begin{bmatrix} a_t \\ s_t \end{bmatrix}^T \mathbf{M} \begin{bmatrix} a_t \\ s_t \end{bmatrix}, t < t_f, \tag{3.2}$$

is a quadratic function that penalizes the dynamics of the system/control at each time step. We also incorporate a final state quadratic cost that penalizes the final state, s_{t_f}, from deviating far from the desired goal g characterized by state s_g,

$$cost(s_{t_f}) = (s_{t_f} - s_g)^T \mathbf{M_f} (s_{t_f} - s_g), \tag{3.3}$$

where $\mathbf{M}$ and $\mathbf{M_f}$ are cost parameters. Similar to the MDP setting, the time-invariant state-action cost function and the final state cost can vary depending on the goal being pursued.

For the discrete MDP setting and the continuous LQR setting, the goal prediction task is defined as follows.

Definition 3.2.1. *The* ***goal prediction task*** *seeks a probability distribution over potential goals given a partial sequence of states:* $P(s_{t_f} = g|s_1, \ldots, s_t)$ *for discrete decision processes and* $P(G = g|s_1, \ldots, s_t)$ *for continuous control processes. In the discrete setting, the exact goal state is reached whereas the final state need only be sufficiently close to state g in the continuous setting.*

3.2.2 Existing Goal Prediction Methods

Many goal prediction methods approach the goal prediction task using Bayesian reasoning. Given a generative, goal-conditioned model of the state sequence, $P(s_{1:t}|g)$, the goal posterior is obtained using Bayes theorem:

$$P_\theta(g_i|s_{1:t_i}) = \frac{P(s_{1:t_i}|g_i)P(g_i)}{\sum_{g' \in \mathcal{G}} P(s_{1:t_i}|g')P(g')}, \tag{3.4}$$

where $s_{1:t_i}$ is the partial trajectory of states from time step 1 to time step t_i, g_i is the inferred goal, and g' of a goal from the set of pre-defined goals ($\mathcal{G}$) in the environment.

A simple Bayesian approach for the discrete setting is the goal-conditioned Markov model (53; 54). It estimates the next state given the current state and goal based on the empirical frequency,

$$P(s_{t+1}|s_t, g) = \frac{\text{count}(s_{t+1}, s_t, g) + \alpha_{s_{t+1}, s_t, g}}{\text{count}(s_t, g) + \alpha_{s_t, g}},$$

where count($\cdot$) is the number of occurrences in the training dataset and α provide a set of optional pseudo-count values. The state trajectory likelihood of (Equation 3.4) is:

$$P(s_{1:t}|g) = P(s_1) \prod_{t=1}^{t_f - 1} P(s_{t+1}|s_t, g).$$

Predestination (55) uses Bayes theorem to infer destinations from driving routes. It uses a history of driver destinations and driving behaviors to predict where the driver is heading (final destination). Similarly, comMotion (56) uses a set of previously visited destinations to predict a person's destination using a Bayes classifier. More sophisticated trajectory likelihood modeling approaches treat the prediction tasks as the "inverse" of a planning process (52; 51; 57; 58). For example, Baker, Tenenbaum & Saxe et al. (52) use inverse planning, which assigns a probability distribution to different plans, to compute goal inferences. They investigated three different settings for goal prediction: single underlying goal, complex goals, and changing goals. In this chapter, we consider the single underlying goal setting and leave extensions to the other settings as future work.

Maximum Entropy Inverse Reinforcement Learning (MaxEnt IRL) (21), which we describe in more detail in the next subsection, uses a trajectory likelihood model to predict driver destinations given partial driving trajectories. It has also been used to predict user intent for robotic teleoperation with application to brain-computer interface (BCI) manipulation tasks (49). In another extension of MaxEnt IRL, the notion of legibility and predictability (51) are used to interpret action analogy, and by Holladay et al (57) to generate pointing configurations that make the goal object legible.

All of the these works use generative models of the trajectory distribution to enable goal prediction using Bayesian reasoning. Though less prevalent, there is some work on discriminative approaches for goal prediction given partial trajectory (59; 60). The Delphian Desktop (60) predicts user intentions in a desktop environment given the cursor trajectories using simple linear regression based on features like peak velocity and distance to the target. Logistic regression (61) has been used to predict the goal given partial trajectory (62) based on features like cosine of the angle, distance using peak velocity, and curve fitting for predicting intended goal. Additionally, the anticipatory temporal conditional random field (ATCRF) (63) and object affordances (23) is used to anticipate human activities. They have produced the Cornell Activity Dataset (CAD-120) (23) for their experiments. We use the same CAD-120 dataset for evaluating our method and compare against this approach.

3.2.3 Maximum Entropy Inverse Optimal Control

Inverse optimal control (also known as inverse reinforcement learning) (64; 65; 66) considers a Markov decision process *without* a reward function and learns the reward function that rationalizes demonstrated decision sequences (66). Assumption a reward function linear in the state feature vectors parameterized by reward parameter θ, $R(s_t) = \theta \cdot \phi(s_t)$, Abbeel & Ng et al. (66) propose the apprenticeship learning approach based on Inverse Reinforcement Learning (65). They devise a strategy of matching feature expectations between expert's policy (π_E) and learner's policy ($\tilde{\pi}$):

$$\left\| \mathbb{E}\left[\sum_{t=1}^{t_f} \phi(S_t) \middle| \pi_E \right] - \mathbb{E}\left[\sum_{t=1}^{t_f} \phi(S_t) \middle| \tilde{\pi} \right] \right\|_\infty \leq \epsilon, \tag{3.5}$$

where ϵ is the largest error allowed when approximately matching feature vectors. While useful for prescriptive behavior in imitation learning tasks, this approach is not as useful for prediction due to the ambiguities arising from many different mixtures of deterministic policies producing the same feature counts.

Ziebart et al. (21) employed the principle of maximum entropy (67) to resolve the ambiguity of mixing policies to match feature counts by selecting a probability distribution:

$$P_\theta(s_{1:t_f}) = \frac{e^{\sum_{t=1}^{t_f} \theta^T \phi(s_t)}}{Z^\theta} \tag{3.6}$$

where $Z^\theta = \sum_{s'_{1:t_f}} e^{\sum_t \theta^T \cdot \phi(s'_t)}$ is the partition function and $s_{1:t_f}$ is the trajectory or path traveled from time step 1 through t_f. The parameters θ that maximize the trajectory log likelihood,

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{s_{1:t_f} \in \Xi} \log P(s_{1:t_f} | \theta), \tag{3.7}$$

are employed by the model. Further, the gradient of Z^θ (partition function) is established in Lemma 3.2.2.

Lemma 3.2.2. *The gradient of the partition function, Z_θ, is:*

$$\nabla_\theta \log Z^\theta_{s_{a\to b}} = -\mathbb{E}\Big[\sum_{t=1}^{t_f} \phi(S_t)|S_1 = a, S_{t_f} = b\Big] = -\mathbb{E}\Big[\phi(S_{a\to b})\Big].$$

Proof. Using the definition of Z^θ from Equation 3.6 we have,

$$\begin{aligned}
\nabla_\theta \log Z^\theta_{s_{a\to b}} &= \frac{1}{Z^\theta_{s_{a\to b}}} \sum_{s_{1:t_f}:s_1=a,s_{t_f}=b} e^{-cost_\theta(s_{a\to b})}(-\phi(s_{a\to b})) \\
&= -\sum_{s_{1:t_f}:s_1=a,s_{t_f}=b} P(s_{a\to b})\phi(s_{a\to b}) \\
&= -\mathbb{E}\left[\sum_{t=1}^{t_f} \phi(S_t)|S_1 = a, S_{t_f} = b\right] = -\mathbb{E}\Big[\phi(S_{a\to b})\Big].
\end{aligned}$$

□

Following from Lemma 3.2.2, the gradient of the trajectory log likelihood function for a set of trajectories and corresponding goals, denoted by Ξ, is:

$$\nabla_\theta \log \prod_{(s_{1:t_f},g)\in\Xi} P(s_{1:t_f}|\theta, g) = \mathbb{E}\left[\phi(S_{1:t_f})|g\right] - \phi(s_{1:t_f}). \tag{3.8}$$

Thus, when maximized, this gradient is zero and the expected feature counts must match the training data feature counts.

In this chapter, we extend MaxEnt IRL to predict human intentions given partial trajectory by maximizing the true goal likelihood instead of the trajectory likelihood.

3.2.4 Inverse Linear-Quadratic-Regulation

Maximum entropy inverse reinforcement learning methods for MDPs have been extended to linear-quadratic regulation (LQR) settings to learn the $\mathbf{M}$ and $\mathbf{M_f}$ coefficient matrices (reward parameters) from demonstrated behaviors using the principle of maximum causal entropy (68). Under this model, computing the features ϕ_{g_i} of the partial trajectory ($s_{1:t_i}$) given the goal (g_i),

$$\phi_{g_i}(s_{1:t_i}) = \sum_{t=0}^{t_i-1} \begin{bmatrix} a_t \\ s_t \end{bmatrix} \begin{bmatrix} a_t \\ s_t \end{bmatrix}^T \tag{3.9}$$

the expectation of the features $\phi_{g_i}(s_{t_i \to g_i})$ of the remaining trajectory ($s_{t_i \to g_i}$) from the current position (t_i) to the goal (g_i),

$$\mathbb{E}[\phi_{g_i}(S_{t_i \to g_i})|g_i] = \sum_{t=t_i}^{t_f-1} \left(\mu_{a_t s_t} \mu_{a_t s_t}^T + \Sigma_{a_t s_t}\right), \tag{3.10}$$

the expectation of the features $\phi_{g_i}(s_{1 \to g_i})$ of the complete trajectory ($S_{1 \to g_i}$) from the starting point to the goal (g_i),

$$\mathbb{E}[\phi_{g_i}(S_{1 \to g_i})|g_i] = \sum_{t=1}^{t_f-1} \left(\mu_{a_t s_t} \mu_{a_t s_t}^T + \Sigma_{a_t s_t}\right), \tag{3.11}$$

can be achieved efficiently based on the fact that all marginal state probabilities are multivariate Gaussians with analytical expressions mean ($\mu_{a_t s_t}$) and variance ($\Sigma_{a_t s_t}$) for these expectations. Finally, the probability of the true goal (g_i) given the partial trajectory ($s_{1:t_i}$) is obtained using Bayes theorem as;

$$P(g_i|s_{1:t_i}) \propto P(g_i|s_t) \prod_{i=1}^{t_f} \pi(a_t|s_t, g_i). \tag{3.12}$$

This predictive linear-quadratic regulator (22) for inverse optimal control is used to predict human intentions and trajectory forecasting. Promising results have been demonstrated on the Cornell Activity Dataset (CAD-120) (23). In this chapter, we have extended the technique used in (22) by training the MaxEnt IRL model by maximizing true goal likelihood.

3.3 Approach

Maximum Entropy Inverse Reinforcement Learning (MaxEnt IRL) is a widely used method to infer the true goal or intentions of a sequential decision maker given a partial trajectory by employing Bayesian reasoning. The reward parameters in the MaxEnt IRL setting are trained via maximizing the trajectory likelihood as shown in Equation 3.8. The trajectory likelihood models are designed and optimized solely with consideration to trajectory prediction rather than goal predictions. While Bayes theorem, the foundation of Bayesian reasoning, holds correctly for the true distributions of goal posteriors and trajectory likelihoods, it can produce error-prone goal posteriors when the likelihood model noisily estimated from limited amounts of available data. To address this problem, in this section, we develop our approach for training the MaxEnt IRL model for goal prediction using goal likelihood maximization in place of the traditional trajectory likelihood maximization approach.

3.3.1 Goal Likelihood Maximization Formulation

To derive our optimization procedure, we first establish Lemma 3.3.1 for computing the gradient (∇_θ) of the log likelihood of a partial trajectory given a goal ($P_\theta(s_{1:t_i}|g_i)$) with respect to the reward parameter (θ).

Lemma 3.3.1. *The gradient for computing the probability of a partial trajectory ($s_{1:t_i}$) given the goal (g_i) can be separated into the sum of expectations and the feature vector,*

$$\nabla_\theta \log P_\theta(s_{1:t_i}|g_i) = -\,\phi_{g_i}(s_{1:t_i}) - \mathbb{E}\Big[\phi_{g_i}(S_{t_i \to g_i})|g_i\Big] + \mathbb{E}\Big[\phi_{g_i}(S_{1 \to g_i})|g_i\Big].$$

Proof. Using the definition from Equation 3.6, we have:

$$\log P(s_{1:t_i}|g_i) = -cost_\theta(s_{1:t_i}) + \log(Z^\theta_{s_{t_i \to g_i}}) - \log(Z^\theta_{s_{1 \to g_i}})$$

[Since, $\log \frac{m}{n} = \log m - \log n$]. Taking the gradient with respect to the reward parameter θ and simplifying after using Lemma 3.2.2 proves Lemma 3.3.1. □

Next, using Lemma 3.3.1, we establish the maximum goal likelihood gradient for MaxEnt IRL given a partial sequence of states.

Theorem 3.3.2. *The gradient for MaxEnt IRL for maximum goal likelihood given a partial trajectory decomposes into a sum of expectations, features and probabilities,*

$$\nabla_\theta \log P_\theta(g_i|s_{1:t_i}) = -\phi_{g_i}(s_{1:t_i}) - \mathbb{E}\Big[\phi_{g_i}(S_{t_i \to g_i})|g_i\Big] + \mathbb{E}\Big[\phi_{g_i}(S_{1 \to g_i})|g_i\Big]$$
$$+ \sum_{g' \in \mathcal{G}} P(g'|s_{1:t_i})\Big(\phi_{g'}(s_{1:t_i}) + \mathbb{E}\Big[\phi_{g'}(S_{t_i \to g'})|g'\Big] - \mathbb{E}\Big[\phi_{g'}(S_{1 \to g'})|g'\Big]\Big),$$

where: $s_{1:t_i}$ is the partial trajectory from time step 1 to t_i, g_i is the true goal and g' are the possible goals ($\mathcal{G}$) in the environment.

Proof. Taking the gradient with respect to θ of the goal log likelihood, after expanding using Equation 3.4:

$$\nabla_\theta\Big(\log P_\theta(s_{1:t_i}|g_i) + \log P(g_i) - \log \sum_{g' \in \mathcal{G}} P_\theta(s_{1:t_i}|g')P(g')\Big)$$
$$\overset{(a)}{=} \nabla_\theta \log P_\theta(s_{1:t_i}|g_i) + \nabla_\theta \log P(g_i) - \log \sum_{g' \in \mathcal{G}} P_\theta(g'|s_{1:t_i}) \nabla_\theta P_\theta(s_{1:t_i}|g')P(g')$$
$$\overset{(b)}{=} -\phi_{g_i}(s_{1:t_i}) - \mathbb{E}\Big[\phi_{g_i}(S_{t_i \to g_i})|g_i\Big] + \mathbb{E}\Big[\phi_{g_i}(S_{1 \to g_i})|g_i\Big]$$
$$-\sum_{g' \in \mathcal{G}} P(g'|s_{1:t_i})\Big(- \phi_{g'}(s_{1:t_i}) - \mathbb{E}\Big[\phi_{g'}(S_{t_i \to g'})|g'\Big] + \mathbb{E}\Big[\phi_{g'}(S_{1 \to g'})|g'\Big]\Big),$$

where: (a) follows from properties of the gradient applied to logarithms and the definition of the goal posterior and (b) is obtained after employing Lemma 3.3.1. □

This gradient trivially equals zero when the goal predictions are perfectly correct (i.e., $P(t_f = g_i|s_{1:t_i}) = 1$). However, this is often difficult to achieve when training from noisy data. In general, optimizing the reward parameter to maximize goal likelihood is non-concave. However, we can obtain a reasonable local maxima by changing the starting conditions and other factors. For example, initializing the reward parameter optimization at the maximum trajectory likelihood parameters guarantees no worse parameters than the trajectory-based approach.

Algorithm 1 Learning IOC model for goal prediction

Input: The reward parameter θ; Set of training trajectories reaching goals Ξ; Set of Goals $\mathcal{G}$
Output: The optimized/learned reward parameter θ

1: **for** (s, g_i, t_i) $\in \Xi$ **do**
2:     Extract partial trajectory $s_{1:t_i}$
3:     $\nabla_\theta \leftarrow -\phi_{g_i}(s_{1:t_i}) - \mathbb{E}\Big[\phi_{g_i}(S_{t_i \to g_i})|g_i\Big] + \mathbb{E}\Big[\phi_{g_i}(S_{1 \to g_i})|g_i\Big]$
4:     **for** g'$\in \mathcal{G}$ **do**
5:         $\nabla_{g'} \leftarrow \phi_{g'}(s_{1:t_i}) + \mathbb{E}\Big[\phi_{g'}(S_{t_i \to g'})|g'\Big] - \mathbb{E}\Big[\phi_{g'}(S_{1 \to g'})|g'\Big]$
6:         Compute $P(g'|s_{1:t_i})$
7:         $\nabla_\theta \leftarrow \nabla_\theta + P(g'|s_{1:t_i})\nabla_{g'}$
8:     **end for**
9:     $\theta \leftarrow \theta + \eta\nabla_\theta$
10: **end for**
11: **return** θ

The learning procedure (Algorithm 1) takes as input an initial reward parameter, a set of training trajectories, and a set of possible goals in the space. It iterates over randomly selected training trajectories, extracting the partial trajectory from the selected trajectory, i.e., $s_{1:t_i}$, and then constructs the full gradient from its components in step 3, step 5, and step 7. Step 3 computes the difference in expected features for the true goal. Step 5 computes the same differences for each possible goal and then Step 7 weights these by the goal probabilities.

Lastly, Step 9 applies a gradient step weighted by η to improve towards locally optimal reward parameters θ^* using expectations computed for the true goal and all other goals. In practice, more sophisticated gradient-based updates (69; 70) can be employed. The algorithm repeats steps 2 through 8 (with decreasing learning weights) for all of the training trajectories until approximately converging to a locally optimal point.

We note the contrast from previous goal prediction methods using MaxEnt IRL trained by maximizing the likelihood over the trajectory to train the reward parameter as explained in (Equation 3.8). Critically, the likelihood of the correct goal given a partial sequence of actions is inferred using Bayesian reasoning. This produces a mismatch between the training and application objective and can produce error-prone goal likelihoods. Thus, the most significant advantage of training using the proposed method (maximum goal likelihood) is that we maximize the likelihood over the true goal, which correctly matches the application objective.

3.3.2 Extension to Linear-Quadratic Regulation

Algorithm 1 provides a general algorithm for MaxEnt IRL trained to maximize goal prediction for the case of discrete state/action decision processes. We can extend this general method to other settings/controllers to match other real-life scenarios. In this chapter, we use inverse LQR to conduct our experiments and we have the cost function $\mathbf{M}$ and $\mathbf{M_f}$ to train as mentioned in section 3.2. In algorithm 1, for our inverse LQR setting we replace the reward parameter θ with $\mathbf{M}$ and $\mathbf{M_f}$. The computation of terms of Algorithm 1 in inverse LQR formulations can be referred from Equations (Equation 3.9), (Equation 3.10), (Equation 3.11), and (Equation 3.12) from section 3.2.4.

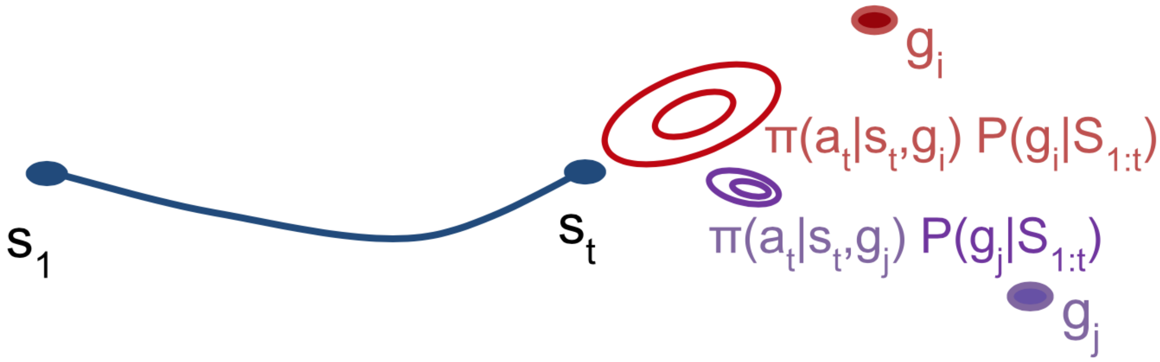


Figure 8. A partial trajectory ($S_{1:t_i}$) and distribution for two goals in the space at a trajectory point S_t. Inverse LQR with the trained reward parameters using the algorithm 1 are used to calculate the distributions for both goals.

Figure 8 depicts the scenario of goal prediction based on the partial trajectory traveled in a real-time situation. There is an agent who starts from the starting point s_1 and travels to point s_t. The goal set $\mathcal{G}$ consists of two goals: g_i and g_j. At trajectory point s_t, we can compute goal distributions for both goals in the space based on the partial trajectory $s_{1:t}$ covered. The color contours represent the corresponding probability distribution (likelihood) of the goal. The corresponding mathematical expressions for each action conditioned on goal in the figure provide the goal probability in the inverse LQR setting, as part of Equation 3.12. These distributions are calculated using the trained reward parameter from Algorithm 1. The most probable goal can be obtained from the posterior goal distribution. In Figure 8, for example, goal g_i is the most probable. Thus, in this way we predict the goal given the partial trajectory in real-time.

3.3.3 Complexity Analysis

The time complexity of the discrete case proposed in Theorem I is $\mathcal{O}(|\mathcal{G}||\mathcal{S}||\mathcal{A}|T)$, where $\mathcal{G}$ is the set of potential goals, $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of action and T is the total time steps in the trajectory (i.e., trajectory length). In this chapter, we have implemented the above general algorithm for the inverse LQR setting which is an example of the continuous case. So, the time complexity of the proposed algorithm for inverse LQR setting requires $\mathcal{O}(T)$ matrix updates.

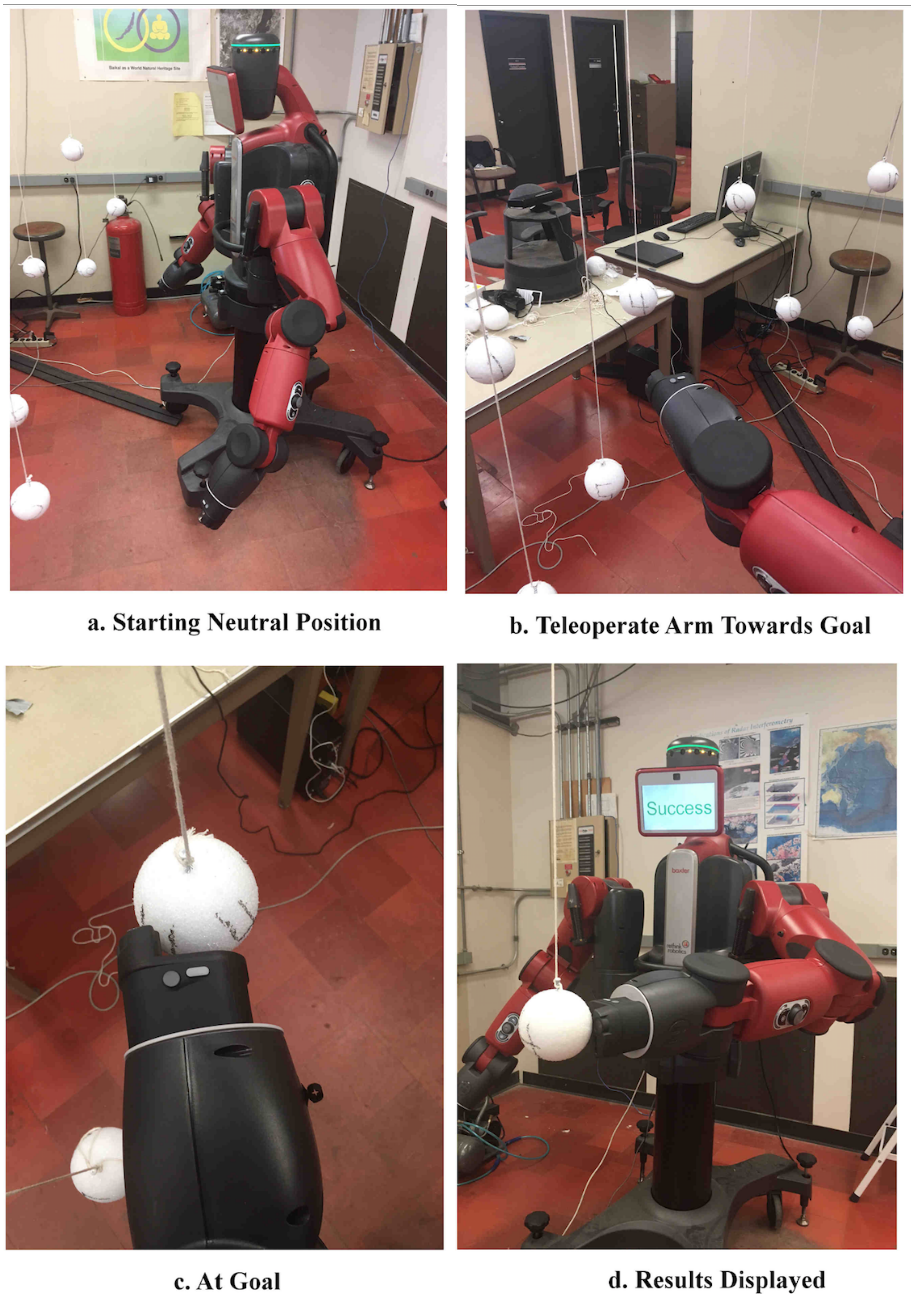
The most significant advantage of using this approach is that the matrix updates only need to be computed once when performing inference over sequences sharing the same time horizon and goal positions. Further, to improve the efficiency of our computation we used the Armadillo C++ linear algebra library for fast linear computations (71).

3.4 Experimental Setup

In this section, we explain our experimental setup used for evaluating our proposed Algorithm 1 from Section 3.3 for the inverse LQR setting. We have used two real-life datasets to evaluate our proposed method.

3.4.1 Goal Pointing Task Data

For our first set of experiments, we have used an existing dataset of pointing tasks (3). The data was collected using a Baxter robot from Rethink Robotics and a Microsoft Kinect camera. For the training data, 10 balls were hung from the ceiling (5 on both sides of the Baxter



a. Starting Neutral Position

b. Teleoperate Arm Towards Goal

c. At Goal

d. Results Displayed

Figure 9. The steps of a task in our testing sequence from a pointing dataset (3) include starting from the robot's neutral position (a) and then teleoperating the arm of the robot (b) to the goal location (c) at which point confirmation is displayed on the robot's screen (d).

robot), and a teleoperator was asked to stand in front of the Kinect Camera (input sensor). The teleoperator was asked to reach the displayed ball number on Baxter's head-mounted display from a neutral position. Another operator moved Baxter's corresponding arm in zero gravity mode from a neutral position to the displayed goal in synchronization with the human arm motion. This Kinect-Baxter correspondence data was used to train a linear regression correspondence model for robotic teleoperation. Further, we used the training sequence to extract states and actions for inverse LQR system and trained cost functions $\mathbf{M}$ and $\mathbf{M_f}$.

The 10 hanging balls from the ceiling were then shuffled to new positions (different from the training set-up) for the testing phase. The 18 teleoperators were asked to teleoperate the Baxter robot's arm by standing in front of the Kinect camera from a neutral position to reach the goal that was displayed on the Baxter head-mounted screen. This process was repeated for each goal and three different control assistance method ((i) Sigmoid assist, (ii) Step assist, and (iii) No assist), for details please refer to (3). The three control assistance methods were also repeated twice in random order to maintain consistency. Thus, each person performed 60 trajectory sequences of reaching the displayed goal.

In total, the dataset consisted of 1080 goal reaching trajectories. Figure 9 explains the steps of test data collection. The dataset contains the Kinect skeleton values, the Baxter end-effector position while the volunteer was teleoperating the Baxter robot and the probability distribution across all five goals along the trajectory. In this chapter, we use the Baxter end-effector positions as the trajectory points (states) for training and testing of the inverse LQR model.

3.4.2 Cornell Activity Dataset (CAD-120)

For our second set of experiments, we employed our Algorithm 1 to train reward parameters on the publicly available Cornell Activity Dataset (CAD-120) to strengthen our claim. This dataset consists of 120 depth camera video of daily activities. There are ten high-level activities: making cereal, taking medicine, stacking objects, unstacking objects, microwaving food, picking objects, cleaning objects, carrying food, organizing objects and eating a meal. These activities are further divided into ten sub-activities: reaching, moving, pouring, eating, drinking, opening, placing, closing, cleaning and null. For example, the task of making cereal can be broken down: reaching (cereal box), moving (cereal box on top of bowl), pouring (from cereal box to a bowl), moving (cereal box to the previous position) and null (moving the hand back).

In this study, we have divided the trajectories based on the above 10 sub-activities. We disregarded null sub-activity as it has an undefined goal or intention. First, we extracted goals for each of the trajectory in the sub-activity. Second, we trained the cost functions $\mathbf{M}$ and $\mathbf{M_f}$ for each of these sub-activities separately. We withheld 10% of each sub-activity dataset for testing and used the rest 90% to train the reward parameters (i.e., $\mathbf{M}$ and $\mathbf{M_f}$). Similar to the previous experiment, trajectory points were used as states and final trajectory point as goal state.

3.4.3 Estimating the Reward Parameters

The inverse LQR model used in this chapter has two separate reward/cost parameter matrices $\mathbf{M}$ and $\mathbf{M_f}$ to train. To provide the strongest guarantees, we first train the reward parameters using the maximum trajectory likelihood method as explained in (Equation 3.8) on the training data for both datasets. Then we use these trained reward parameters to initialize Algorithm 1 to learn using our proposed method for maximizing goal likelihood on the training data.

We have used accelerated stochastic gradient descent with an adaptive learning rate (69; 70) and L1 regularization on both parameters simultaneously. This regularized approach prevents over-fitting over the demonstrated trajectories of the datasets used in this chapter. In the next section, we would describe goal predictions using inverse LQR controller on the test data for both datasets.

3.4.4 Goal Prediction via Inverse LQR

Following the existing formulations employed for maximum trajectory likelihood methods (3), a goal is defined as a location in x_g, y_g, z_g translational space that we want the robot arm end-effector to approximately reach. The end-effector is the endpoint of the robot arm, which is calculated using forward kinematics (72). The end-effector consists of x_t, y_t, z_t translational and x_r, y_r, z_r, w_r quaternion angles as rotational dimensions referenced from the associated robot's coordinate frame. We have considered only translational dimensions for goal positions.

Following the approach outlined for the inverse LQR setting (22), the authors of (3) assume the linear dynamics of (Equation 3.1), in which the state of the end-effector is defined as,

$$s_t = [x_t, y_t, z_t, \dot{x}_t, \dot{y}_t, \dot{z}_t, \ddot{x}_t, \ddot{y}_t, \ddot{z}_t, 1]^T, \tag{3.13}$$

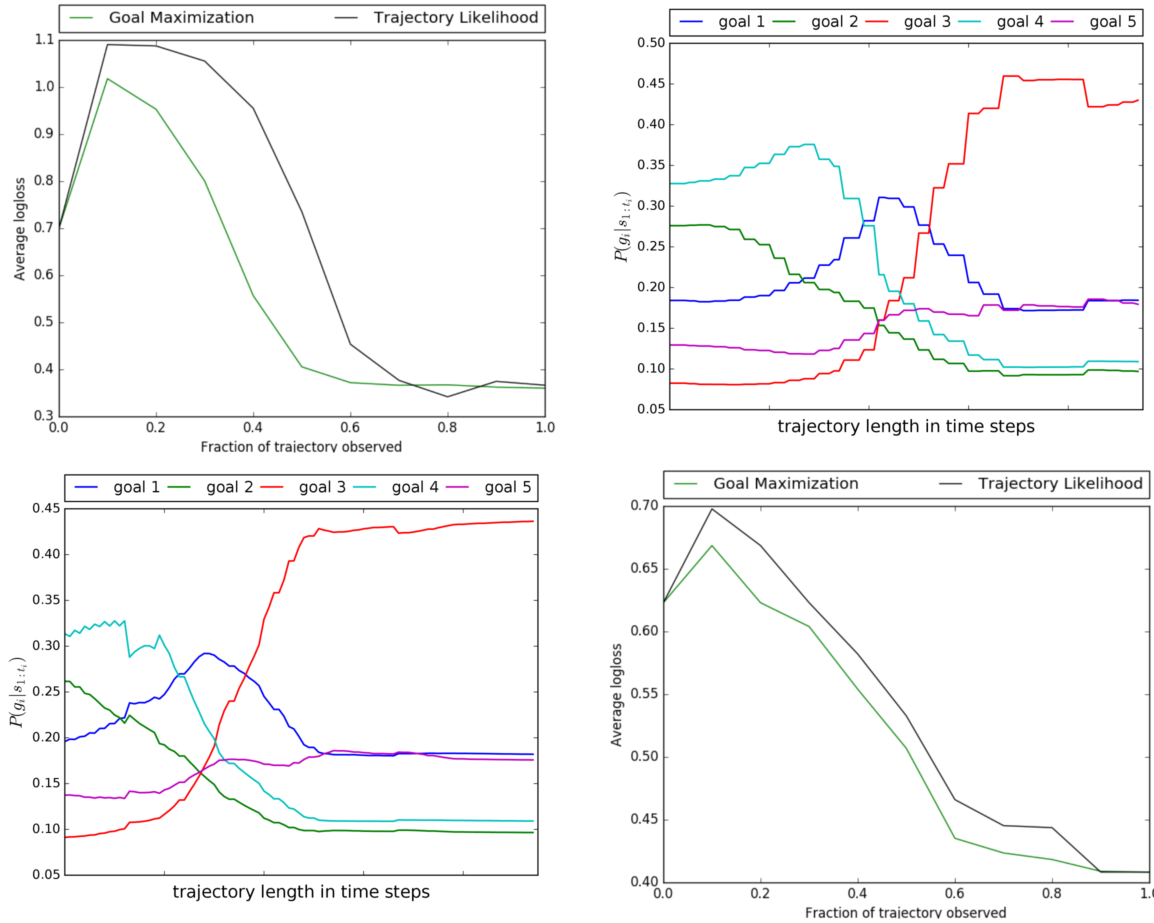


Figure 10. (a) Plot showing comparison of logloss by our goal likelihood maximization method to the trajectory likelihood maximization model on goal pointing task data; (b) Change of probability distribution over goals across a trajectory of reaching goal #3 using trajectory likelihood maximization model; (c) Change of probability distribution over goals across a trajectory of reaching goal #3 using goal likelihood maximization method; (d) Plot showing comparison of logloss by goal likelihood to trajectory likelihood on CAD-120.

and end-effector actions as

$$a_t = [\dot{x}_t, \dot{y}_t, \dot{z}_t]^T, \tag{3.14}$$

where $(\dot{x}_t, \dot{y}_t, \dot{z}_t)$ are velocities, $(\ddot{x}_t, \ddot{y}_t, \ddot{z}_t)$ are accelerations, and a constant of 1 is added to the state representation to incorporate linear features into the quadratic cost function in (Equation 3.2). Additionally, goal state i of the end-effector is represented using only the goal's translational position,

$$g_i = [x_{g_i}, y_{g_i}, z_{g_i}, 0, 0, 0, 0, 0, 0, 0]^T. \tag{3.15}$$

To compute goal predictions along the test trajectories, we train the reward parameters $\mathbf{M}$ and $\mathbf{M_f}$ using our proposed method (maximum goal likelihood) as described in Algorithm 1 on the training data. From these trained cost matrices, the probabilities of different possible goal states are inferred given the observed partial trajectory of the end-effector in real time. The process is clearly depicted in Figure 8 and (Equation 3.12). These goal state probabilities are $P(g_i|s_{1:t_i})$ and the probability of the most likely intended goal of the partial trajectory, I, is,

$$I = \max_i P(g_i|s_{1:t_i}). \tag{3.16}$$

3.4.5 Prior Distribution

The inverse LQR goal prediction method is a Bayesian inference method that benefits significantly from a prior distribution over the possible goals (22). In the previous trajectory likelihood maximization experiment (3), they used a distance prior similar to the one used in previous work (22),

$$P(g_i|s_t) \propto e^{-\beta dist(s_t, g_i)}, \tag{3.17}$$

where $dist(s_t, g_i)$ is a function that computes the Euclidean distance between the spatial coordinates of s_t and g_i, and β is an adjustable coefficient that increases the importance of distance on the distribution. As $dist(s_t, g_i)$ decreases, $P(g_i|s_t)$ increases effectively making closer targets more probable. We have used the same formulation for most intended goal prediction for both experiments in this chapter.

3.4.6 Baselines

To compare our goal likelihood method on goal pointing task data from the two datasets, we use the nearest target (predicting the nearest goal as the true goal along the trajectory points) prediction. It is the simplest baseline for goal prediction, and all methods should be expected to perform better than it. We additionally use logistic regression (62) as the discriminative method comparison baseline. Also, we compare with the previous approach of constructing a model using trajectory likelihood maximization (22). For CAD-120 dataset, in addition to comparing to the trajectory likelihood method, we also compared our method with ATCRF (63).

3.4.7 Evaluation Metrics

To evaluate our proposed method against the existing trajectory maximum likelihood method, we use two evaluation metrics. First, we compute the logarithmic loss for true goal probability

TABLE V

A COMPARISON OF THE TRAJECTORY LIKELIHOOD MODEL, THE GOAL LIKELIHOOD MODEL, AND THE NEAREST GOAL BASELINE FOR THE GOAL POINTING TASK DATASET EVALUATED USING THE ACCURACY, MACRO PRECISION, AND MACRO RECALL GIVEN VARIOUS FRACTIONS OF THE TRAJECTORY.

		Fraction of the trajectory				
Method	Measure	20%	40%	60%	80%	100%
Nearest Goal	Accuracy	21.4	28.6	58.6	100	100
	Macro Prec.	50.0	50.0	50.0	100	100
	Macro Recall	10.7	14.2	39.3	100	100
Trajectory Likelihood	Accuracy	28.6	28.6	64.3	100	100
	Macro Prec.	50.0	50.0	50.0	100	100
	Macro Recall	14.3	14.3	32.2	100	100
Goal Likelihood	Accuracy	28.6	35.7	92.8	100	100
	Macro Prec.	50.0	50.0	50.0	100	100
	Macro Recall	14.3	17.9	46.5	100	100

across the whole trajectory. The logarithm loss has been plotted for both methods at various fractions of the trajectory covered in Figure 10-a on pointing task dataset and Figure 10-d on CAD-120. Second, we compute the accuracy of our proposed method and other baselines across different fractions of the trajectory in predicting the true goal. We have also reported precision and recall for both methods. Table V and Table VI report the results for both datasets.

3.5 Results and Discussion

The proposed optimization of the reward parameter to maximize goal likelihood involves maximizing a non-concave function. This prevents any guarantees of convergence to a global optimum. However, still, we can reach some local maximum that provides a better result than previous trajectory-based optimization methods. We have experimented with three different

TABLE VI

A COMPARISON OF THE TRAJECTORY LIKELIHOOD MODEL, THE GOAL LIKELIHOOD MODEL, AND THE ATCRF MODEL FOR THE CAD-120 DATASET EVALUATED USING ACCURACY, MACRO PRECISION, AND MICRO PRECISION GIVEN VARIOUS FRACTIONS OF THE TRAJECTORY.

		Fraction of the trajectory				
Method	Measure	20%	40%	60%	80%	100%
ATCRF (63)	Accuracy	-	-	-	-	86.0
	Macro Prec.	-	-	-	-	84.2
	Macro Recall	-	-	-	-	76.9
Trajectory Likelihood	Accuracy	80.9	82.5	84.1	90.4	100
	Macro Prec.	65.0	73.4	79.1	87.5	100
	Macro Recall	77.3	91.4	94.2	96.2	100
Goal Likelihood	Accuracy	81.8	86.4	90.1	100	100
	Macro Prec.	71.8	78.1	83.3	100	100
	Macro Recall	75.0	81.0	87.5	100	100

starting points to train the cost function $\mathbf{M}$ and $\mathbf{M_f}$: (1) initial values of all 0; (2) pre-trained initial values using the optimization objective of past work (i.e., trajectory likelihood maximization); and (3) randomized starting points. We find convergence to very similar parameters with all three of the different starting points, indicating that we can reach a stable local maxima without strong sensitivity to the initial values.

We have tested our method on two different real-life datasets involving human and robot goal-directed movements. Figure 10-a illustrates the logarithmic loss of the correct goal prediction given a partial trajectory computed across the fraction of the trajectory for the pointing task dataset. The black color represents the trajectory likelihood method and the green color represents the proposed goal likelihood approach. It is evident from Figure 10-a that the goal likelihood maximization method's logarithmic loss decreases faster and reaches the true goal probability in approximately 50% of the trajectory. On the other hand, the trajectory likelihood maximization method achieves the same performance at 70% of the trajectory. In both settings, we have used a distance prior, so the probability distribution rapidly increases from a uniform distribution as the true goal may be farther from the neutral position than other targets.

To illustrate the behavior of the goal prediction methods, we select a trajectory from pointing task test data and plot the probability distribution across five goals along the trajectory length in Figure 10-b and c. The plot of the resulting distribution in Figure 10-b corresponds to the trajectory likelihood method and Figure 10-c corresponds to our proposed goal likelihood maximization method. We can see that our goal likelihood maximization method performs better than the trajectory likelihood maximization method. Our proposed method realizes a high probability prediction for the true goal much earlier than the previous trajectory likelihood maximization method with a smoother transition across different goal probabilities.

Figure 10-d shows the logarithmic loss of goal prediction along the trajectory for reaching a goal from the CAD-120 dataset. The trajectory likelihood maximization method is represented by the black color and our proposed goal likelihood maximization method is shown in green. The plot clearly shows that our goal likelihood maximization method predicts the true goal (approximately 60%) much earlier in the trajectory than the trajectory likelihood method (approximately 80%).

In Table V, we report the accuracy, precision, and recall for goal prediction for three methods, i.e., the nearest goal predictor, trajectory likelihood maximization model, and the goal likelihood maximization model. Both the previous (trajectory likelihood) and proposed (goal likelihood) models perform significantly better than the simplest baseline method, i.e., the nearest goal baseline. At 40% and 60% of the trajectory, our proposed goal likelihood-based method outperforms the trajectory likelihood-based method by a noticeable margin. The result also matches with our log loss metrics as shown in Figure 10-a. We also compare our results with a logistic regression model (62) as the generative method baseline. The reported goal prediction accuracy of 57.9% is obtained from a partial trajectory of length 60 time-steps. The average range of the trajectories of pointing task dataset is 110 time-step. So, at 60% of the trajectory length, we found that our proposed method predicts the true goal with an accuracy of 92.8%, which is significantly better than logistic regression.

Table VI shows the performance results of the experiment conducted on the CAD-120 dataset. We compared the performance of our proposed goal-based method with other baselines based on trajectory likelihood maximization and the ATCRF model (only result for 100% is available). We can see that the trajectory likelihood method achieves comparable accuracy at 40% of the trajectory what is not realized until 100% of the sequence is observed using the ATCRF model. The result of the ATCRF method is on the unmodified CAD-120 dataset, which consists of null sub-activities, which prevents it from achieving 100% accuracy even when observing the complete trajectory. From the beginning of the trajectory, our proposed goal-based method outperforms the trajectory-based method by a considerable margin, which matches the log loss results shown in Figure 10-d and achieves 100% accuracy in prediction at 80% of the trajectory.

Thus, these experiments strongly support our claim that by re-training the MaxEnt IRL approach using goal likelihood maximization for goal predictions, we can achieve better and faster goal prediction than existing methods—specifically those based on trajectory likelihood maximization. As this is an important subproblem for planning symbiotic robot behavior, we believe these improvements will help increase the productivity of human-robot collaborative tasks when used appropriately.

3.6 Conclusion and Future Work

In this chapter, we have proposed training inverse reinforcement learning models that were initially designed for policy estimation, to instead be optimized for goal prediction. We derived the gradient for optimizing goal likelihoods under the general discrete maximum entropy inverse reinforcement learning (MaxEnt IRL) setting and under the continuous inverse linear-quadratic regulation (LQR) setting. We demonstrated that our goal likelihood maximization method provides significant improvements for goal prediction compared to previous methods based on trajectory likelihood maximization in practice. Thus, with our new approach, we can more accurately infer intended goals farther in advance than previous approaches, enabling robots to know human intentions to make more compatible decisions.

As future work, we will test our method on real-world human-robot tasks like assisting robotic teleoperation (3). These tasks often involve additional complications that should also be modeled to improve goal prediction. For example, though we have assumed that the robot's workspace is free of obstacles in this chapter, many real-world robotic workspaces contain numerous obstacles. We plan to extend our goal prediction optimization approach to the hybrid, two-level imitation learning method (73) that incorporates discrete waypoints at the top level and employed LQR predictions conditioned on the waypoints at the bottom level. We believe that through arm motion demonstrations of obstacle avoidance during training, the cost function can be learned to reason about arm movements around obstacles in testing environments. Further, in this chapter, we have assumed that the goals are static in the environment. We will relax this assumption by allowing goals to change over time without being reached (52; 19).

3.7 Acknowledgment

We thank Mathew Monfort for sharing his linear quadratic regulator for inverse optimal control code with us for modification. This research is supported as part of the Future of Life Institute (futureoflife.org) FLI-RFP-AI1 program, grant #2016-158710 and NSF grant #1652530.

CHAPTER 4

ROBOT LEARNING TO MOP LIKE HUMANS USING VIDEO DEMONSTRATIONS

4.1 Introduction

Many daily tasks such as mopping may seem mundane to humans, but are surprisingly hard to automate effectively and comprehensively. Best in class approaches such as the Roomba provide one-size-fits-all solutions, but struggle with hard floors, grout, situations with space constraints (e.g. bathrooms) and so on. Teacher-student approaches such as imitation learning offer the potential of efficiently teaching specialized heuristics to robots (74; 75; 76; 77; 78; 79; 4), especially if the input problem (demonstrating human teacher behavior) can be addressed at scale. The option of instrumenting humans with sensors (via motion-tracking suits or other sensors) has been pursued as one way to obtain such input. However, sensor-based approaches don't scale well, and alter the natural human behavior we are trying to mimic. Video is a more natural recording mechanism, but it is less precise and typically requires machine learning (ML) to interpret useful signal. This chapter develops and applies imitation learning from video input to stylistically mimic effective human mopping movements by a robotic arm.

Artificial Intelligence has made great progress in imitating human behaviors from video demonstrations like the imitation of pouring tasks using a robotic arm (4). Similarly, a robot can learn a mopping task from a human mopping demonstration in the form of a video. It is

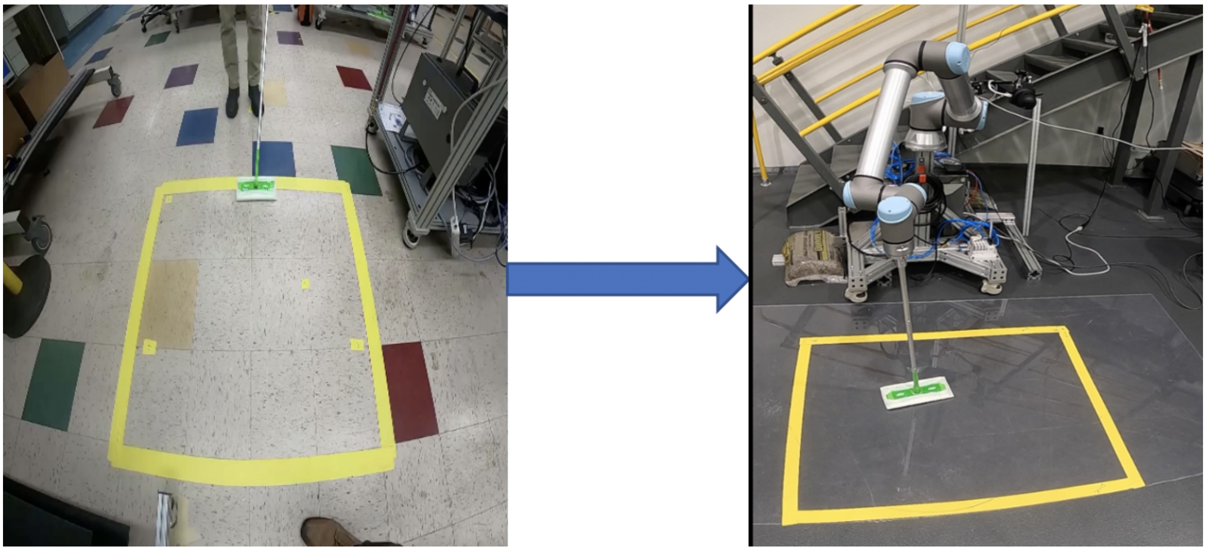


Figure 11. Robot learning to mop from human mopping video demonstration.

very convenient for a human to record a video of themselves mopping the floor, which could be used to teach robots to mop inside a home. There is also an abundance of freely-available videos that can be used as demonstrations to learn mopping skills for a robotic arm.

In this chapter, we learn mopping behavior from human video and imitate it using a robotic arm attached with a mop, as shown in Figure 11. We build two robotic systems to learn and imitate mopping behavior from human demonstration collected using a camera. The first robotic system is a control-based approach that uses a traditional Computer Vision tracking method to track (x, y) position of the mop in human demonstration video and inverse Kinematics to transform the mop (x, y) tracked position to robot joint configurations. Our second robotic system is a learning-based approach that uses an advanced computer vision method, the Time Contrastive Network (TCN) (4) to learn a multi-view (first and third person) TCN embedding on human demonstration videos. The trained TCN model reduces high-level human demonstration videos to a low-level 32-bit embedding per frame to teach the robot to mop. To learn the mopping motion on a robotic arm, we use reinforcement learning (RL), more precisely PILQR, which is a combination of model-based and model-free algorithms (80). We compare the robot's first-person TCN embedding of performing the mopping task with the third-person human demonstration TCN embedding as the reward to RL and update the robotic motion to better imitate the task. For effective learning of the human video motion to the robot using RL, we devised a reward function which is a combination of TCN embedding-based reward and z-axis based reward to constrain the mop to contact the floor at all-time during the mopping task.

There are several approaches to build cleaning robots (81) to wipe, scrub or sweep surfaces using classical control (position control (82; 83; 84; 85; 86; 87; 88; 89), and force control (90; 91; 92; 93; 94; 95)), learning based (supervised learning (96; 97; 98), and learning from demonstration (95; 99; 100; 101; 102; 103), and reinforcement learning (104; 105)). Most of these methods use kinesthetic teaching to move the robot or plan a trajectory across given points for cleaning surfaces but not necessarily imitate human behavior. We are the first to build an end-to-end learning-based robotic system that learns human mopping behavior from human video demonstration. We use a Universal Robots (UR)-10e[1] robotic arm attached to a

---

[1]https://www.universal-robots.com/products/ur10-robot/

Swiffer mop. A first-person camera is placed to the robotic arm to provide the reward to the RL algorithm to correct its robotic arm motion to imitate human motion.

The chapter is organized as follows: We start by describing our two robotic systems to perform mopping by imitating human behaviors from video data. Next, we explain the experimental setup used to evaluate our proposed robotic system. The results section summarizes the results obtained for the two proposed robotic systems. Then, we provide a summary of background information on imitation learning, previous work on imitation learning from videos, and mopping robots. Lastly, we provide conclusions and propose future work.

4.2 Approach

Mopping the floor is an essential daily task. We devised two robotic systems to perform mopping using a robotic arm by imitating human mopping behaviors from videos of the activity. In this section, we explain our two proposed robotic systems. The first robotic system comprises of a traditional computer vision technique for tracking and then uses inverse kinematics to move the robotic arm. The second robotic system uses an advanced computer vision technique, Time Contrastive Networks (TCN) (4) and reinforcement learning to move the robotic arm to mimic human mopping behavior from videos of the activity.

4.2.1 Method 1: Tracking and Inverse Kinematics Approach

The first system is a position control-based robotic system built of traditional Computer Vision tracking method and Inverse Kinematics. This method serve as baseline to our proposed learning-based method. There is a yellow box (3 X 4 ft) made in front of humans where they perform mopping tasks and a similar yellow box is marked in front of the robot mopping area.

One of the ways to track the movement of the mop head is using OpenCV's object tracking functionality. We can localize the position of the mop head inside the yellow square by utilizing knowledge of the real-world coordinates of the corners of the yellow square. Since the video is affected by lens distortion as well as projective transformation (parallel lines not appearing parallel), it is important to correct these factors and get an orthographic view of the yellow square. This can be achieved by computing a perspective transformation matrix. We choose four corner points of the yellow square in a video frame as the boundaries of the new area of interest. Then we map each corner in the frame with a corner in a new square image that is theoretically bounded by the yellow square. For example, the top-left corner of the yellow square becomes the top-left corner of the new image specified as (0, 0). We use these mappings between the corners of the yellow square and the corners of the new image to compute a perspective transformation matrix that is used to map points from the original frame to an approximate orthographic view. Then we can warp the yellow square into the orthographic view using the computed matrix and can perform this operation for every frame since the 3rd person videos involve a static camera in which the position of the yellow square does not change.

Once we have the perspective transformed videos, we track the mop head using the object tracking functionality present in OpenCV. We use the Boosting tracker, which is an online version of the AdaBoost classifier (106), that gets trained to classify the target object as new frames come in. We supply the initial bounding box of the mop head in the first frame and feed it to the tracker which updates the bounding box to the position of the mop head. To track the position of the mop in the real world, we first find out the coordinates of the corners of the yellow square with respect to the robot's coordinate system. Since the new frames are warped using a perspective transformation, these real-world coordinates form the boundaries of all these frames. Then we can derive the real-world coordinates of the mop head by first finding the centroid of the bounding box and then translating the pixel coordinates of the centroid from the top-left corner, to the real-world coordinates.

The fully defined mop head pose is created by using the obtained coordinates (x, y) of the mop head from the human demonstrated video, a fixed z height of the mop on the floor, and a fixed orientation. For each time step, we append the resulting pose to an array of waypoints for the robot to follow. The robot is controlled using ROS Kinetic (Robot Operating System (107)) and a UR 10e robot model with updated kinematics to include a mop handle and mop head-end effector. We publish the waypoint array to the MoveIt [1] Motion Planning Framework to compute a Cartesian path with reference to the mop head using inverse kinematics (108). We start the robot at an initial joint position and restrict the joint limits of the robot arm for each joint angle from +/- 2π to +/- π to resolve the multiple possible joint configuration issues of inverse kinematics. The resulting joint configuration obtained for each time step corresponds to the obtained mop head position (x,y) from the human demonstration video, and we then send the joint angles to the robot to move the arm.

---

[1]https://moveit.ros.org/

4.2.2 Method 2: Time Contrastive Network and Reinforcement Learning Approach

In this subsection, we describe our proposed learning-based robotic system that is a combination of Time Contrastive Networks (TCN) and Reinforcement Learning (RL). We took inspiration from the pouring task presented by Sermant et al 2017 (4) and devised a new reward function to perform the mopping task on a robotic arm from human demonstration provided in the video.

4.2.2.1 Time Contrastive Network

The main goal is to mimic human mopping behavior provided in the video to a robotic arm. The video is a high-level representation of the task. The 32-bit embedding provided from TCN (4) can be a good low-level representation of the video and can be used as a reward to train our robotic arm to mimic human mopping behavior.

In our day-to-day life, we observe something in our third-person view and perform the task in our first-person view. Similar to this analogy, we use a multi-view (first-person and third-person) video of the mopping task performed by humans to train a TCN model. The TCN is a deep neural network architecture comprising of pre-trained InceptionV3 (109) on ImageNet dataset (110) and embedding layer (conv2D, spatial softmax and fully connected layer) as shown in the Figure 12. The positive frame is extracted from one of the view videos and the anchor frame is extracted from the same temporal frame from another view video. A negative frame is extracted from a distance (α) of some frames (one-sec equivalent frames difference) from the positive frame. Figure 13 depicts the all three frames extraction from two views of the task videos. A positive frame, anchor frame, and negative frames are given input to the TCN network. The TCN provides 32-bit embedding of the input frames which is used to compute a loss. The embedding of an input image is 'x' is represented by $f(x) \in \mathbb{R}$. In this case, we use a triplet loss (111; 4) to have positive ($f(x_i^p)$) and anchor ($f(x_i^a)$) frames closer to each other (similar embedding values) and negative frame ($f(x_i^n)$) farther or in a different cluster. The triplet loss can be given as:

$$\left\| f(x_i^a) - f(x_i^p) \right\|_2^2 + \alpha < \left\| f(x_i^a) - f(x_i^n \right\|_2^2, \forall (f(x_i^a), f(x_i^p), f(x_i^n)) \in \mathbb{T} \tag{4.1}$$

where $\mathbb{T}$ is set of all possible triplets in the training set.

4.2.2.2 Reinforcement Learning to learn mopping behavior

The main problem we are trying to solve is to perform Imitation Learning from human mopping video demonstration to a robotic arm attached with a mop. For reinforcement learning, the big challenge is to find a good reward function for the real-life task. We resolved this issue using a TCN based reward function that is obtained from a human video demonstration. We can represent the problem of robot learning for a mopping task as the Markov Decision Process:

- State (x_t): Joint angles of the robot; Joint Velocities of the robot; Robot first-person video TCN embedding; Z-axis of the mop
- Action (u_t): Robot Joint angles ;
- τ is the state transition probability for next state x_{t+1} from state x_t under action u_t;
- Reward ($R(x_t)$) is the reward or cost received by visiting state x_t

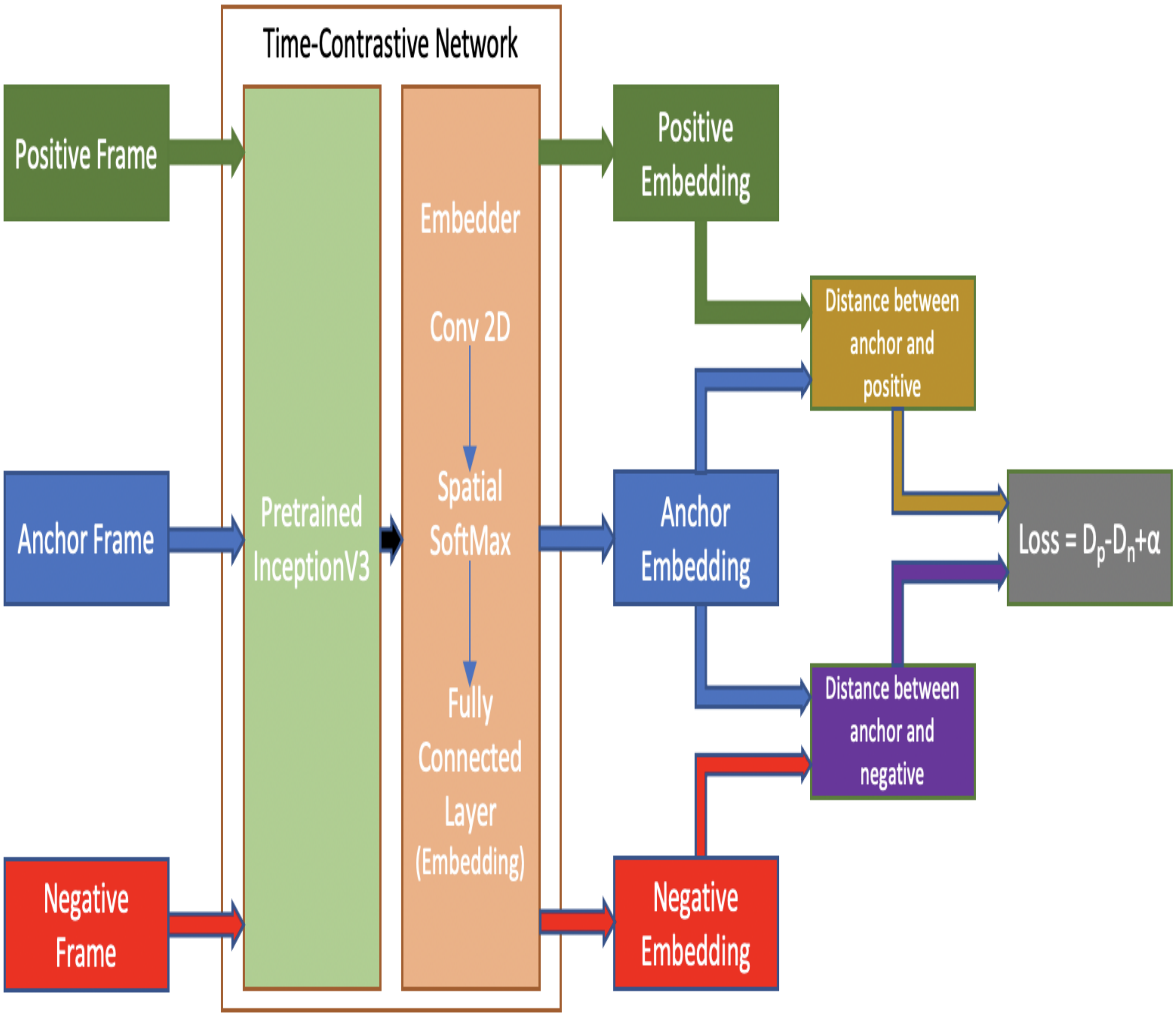


Figure 12. Time Contrastive Network Architecture

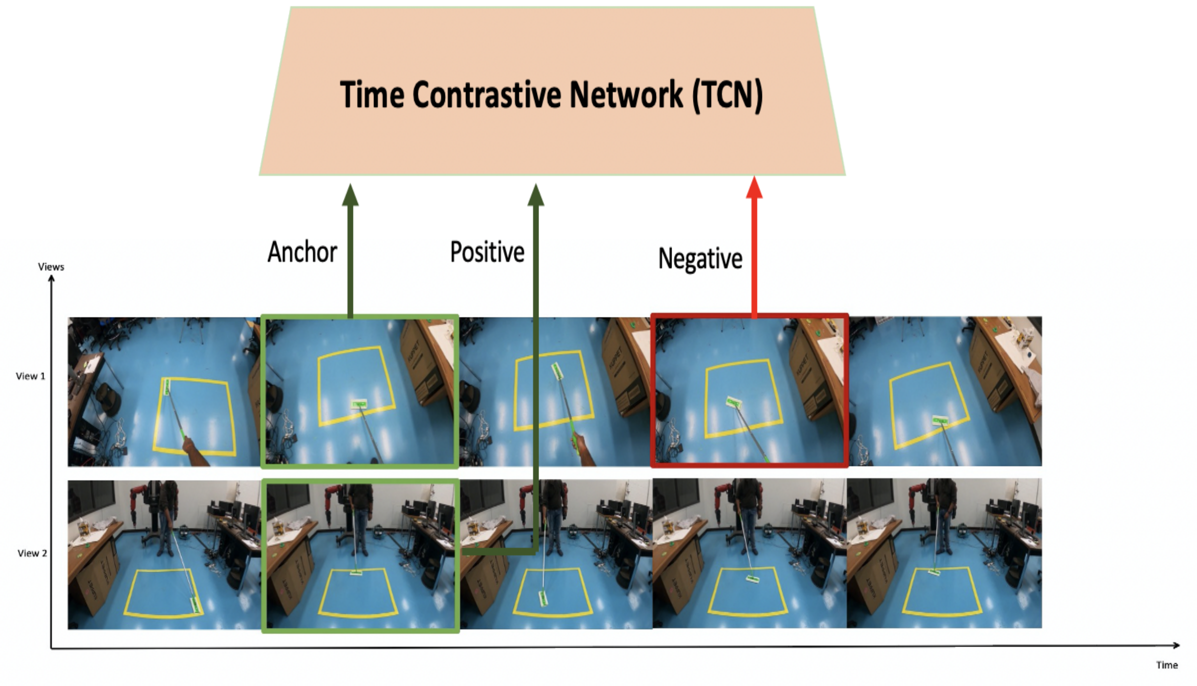


Figure 13. Frames extraction from videos for TCN

The main contribution of this chapter is devising a reward function for a mopping task. We used a mixture of TCN based reward and z-axis of the mop reward. The TCN based reward takes a frame from the human third-person video and a frame from the robot's first-person video of performing the mopping task and passed it to a trained TCN network which outputs 32-bit embedding of both the input frames. The TCN reward as shown in (Equation 4.2) comprises of squared euclidean loss which gives larger gradients when the embeddings are further apart (in our case robot mop is far away from human demonstration mop) and a Huber-style loss to ensure fine-grained movements of the mop with respect to the human demonstration. The TCN loss helps us to correctly position the mop on the floor by imitating human mopping behavior. However, there is a z component of the mop that is difficult to capture from the video demonstration. Also, during our RL training we noticed it was very difficult to imitate mopping behavior using only TCN loss. Thus, we devised a second loss based on the z-axis of the mop as shown in (Equation 4.3). This reward helps us to keep the mop on the floor all the time by subsequently penalizing it for having the mop in the air or going below ground level. We have experimented with different z based rewards, for example linear function etc.

Let T be the total time step of the video sequence; V = $(v_1, \ldots, v_T)$ be the TCN embedding of each frame of the third-person human video demonstration sequence; W = $(w_1, \ldots, w_T)$ be the TCN embedding of each frame of the first-person robot video sequence. Then the TCN reward is given by,

$$R(v_t, w_t) = -\alpha\|w_t - v_t\|_2^2 - \beta\sqrt{\gamma + \|w_t - v_t\|_2^2} \tag{4.2}$$

where α and β are empirically chosen weights for squared euclidean loss and Huber-style loss and γ is a small constant.

$$\mathbf{R}(\mathbf{Z}) = \begin{cases} a_1(b_1 + Z_{value})^2, & \text{if } Z_{value} \geq z_{const} \\ a_2(b_2 + Z_{value})^2, & \text{otherwise} \end{cases} \tag{4.3}$$

where Z_{value} is the value of the robot mop and z_{const} is the constant z value recorded when mop is on the floor. a_1 and b_1 are empirically chosen constants used when the mop is above the ground and a_2 and b_2 for penalizing more if the mop is going beyond the ground level (z_{const}).

We have used PILQR (80) as our reinforcement learning (RL) algorithm which is a combination of model-based and model-free approach. The PILQR method is sample efficient and has good success with real-world robotics tasks. The details of PILQR is explained in the next subsection 4.2.2.3.

4.2.2.3 Reinforcement Learning (PILQR) Details

Let $p(u_t|x_t)$ be a robot policy that defines a probability distribution over robot action u_t conditioned on the state x_t at each time step t of the mopping task trial. We use policy search to optimize the policy parameter θ. Let $\tau = (x_1, u_1, \ldots, x_T, u_T)$ be a trajectory consisting of states and actions. For a given cost function $c(x_t, u_t)$, we can define the trajectory cost as $c(\tau) = \sum_{t=1}^{T} c(x_t, u_t)$. The policy is optimized with respect to the expected cost of the policy

$$J(\theta) = \mathbb{E}_p\left[c(\tau)\right] = \int c(\tau)p(\tau)d\tau\,, \tag{4.4}$$

where $p(\tau)$ is the policy trajectory distribution given the system dynamics $p\left(x_{t+1}|x_t, u_t\right)$

$$p(\tau) = p(x_1)\prod_{t=1}^{T} p\left(x_{t+1}|x_t, u_t\right) p(u_t|x_t)\,. \tag{4.5}$$

In this chapter, we use time-varying linear-Gaussian (TVLG) controller $p(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t x_t + \mathbf{k}_t, \mathbf{\Sigma}_t)$. We apply PILQR(80) reinforcement learning method to learn these TVLG controllers on a real robot. PILQR combines model-based and model-free policy updates for sample-efficient learning of tasks with complex system dynamics.

Let $S(x_t, u_t) = c(x_t, u_t) + \sum_{j=t+1}^{T} c(x_j, u_j)$ be the cost-to-go of a trajectory starting in state x_t by taking an action u_t and following the policy $p(u_t|x_t)$ thereafter. For each iteration i, PILQR performs a KL-constrained optimization of $S(x_t, u_t)$:

$$\min_{p^{(i)}} \; \mathbb{E}_{p^{(i)}}[S(x_t, u_t)] \; s.t. \; \mathbb{E}_{p^{(i-1)}}\left[D_{\mathrm{KL}}\left(p^{(i)} \| \, p^{(i-1)}\right)\right] \leq \epsilon,$$

where limiting the KL-divergence between the new policy $p^{(i)}$ and the old policy $p^{(i-1)}$ leads to a better convergence rate. The optimization is divided into two steps. The first step involves performing a fast model-based update using an algorithm LQR-FLM (112), which is based on the iterative linear-quadratic regulator (iLQR) (113) and approximates $S(\mathbf{x}_t, \mathbf{u}_t)$ with a linear-quadratic cost $\hat{S}(x_t, u_t)$. The second step optimizes the residual cost-to-go $\tilde{S}(x_t, u_t) = S(x_t, u_t) - \hat{S}(x_t, u_t)$ using a model-free method PI^2 (114; 115) to produce an unbiased policy update.

4.3 Experiment

In this section, we explain our human mopping video data collection, data pre-processing & video alignment, TCN training, simulation building, RL training on the real robot mimicking human mopping behavior from video demonstration.

4.3.1 Hardware Used

For human mopping video demonstration data collection, we use two GoPro Hero 8 cameras with 60Hz resolution to record first and third person video of the mopping task. We use a Universal Robot UR10e mounted on a fixed pedestal stand. We 3D printed an attachment to the robot to connect a mop. We used Swiffer mop for our mopping task. We use Amcrest wifi camera on top of the robot to record first-person video.

4.3.2 Data Collection

First, we prepared the work area for data collection. On the floor, we marked an area of 3feet by 4feet with a yellow boundary. One of the cameras was mounted stationary on a benchtop for third person shot and another camera was mounted to a head strap for the first person shot. There are four types of motion that we collected as described in Figure 14. Each demonstrator performed each type of motion twice. There were 15 demonstrators who performed the mopping task using a floor mop.

4.3.3 Data Pre-processing and Video Alignment

The collected video data is high resolution and reduced to 1080p to avoid memory issues during pre-processing and training. The audio is removed to reduce the size of the videos. The

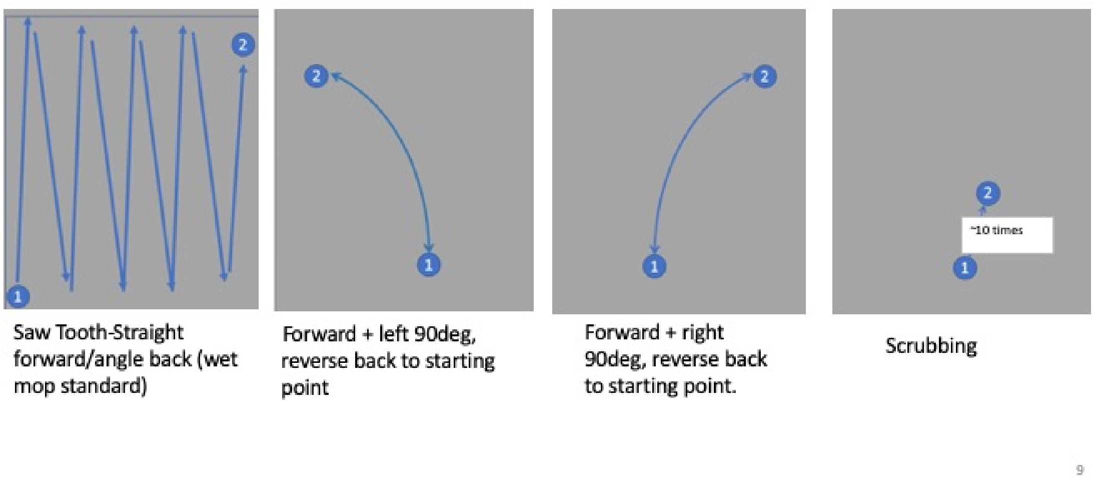


Figure 14. (a) Saw tooth straight forward/angle back, (b)Forward + left 90°, (c) Forward + right 90°, (d) Scrubbing between point 1 and 2

next step in the pipeline is to align first and third-person perspectives with each other which were a strong assumption for TCN training.

Multi-perspective videos are a set of videos captured from different viewpoints by different cameras, each in their own position in space. Often, the cameras are out of sync and start their recordings at different times. This in turn causes the resulting videos to be out of sync as well. Training the Time Contrastive Network (TCN) on unsynchronized videos will make it learn the wrong representations that can later have an adverse impact on training the RL model. Thus, it becomes extremely important to perform video synchronization across all the perspectives before initiating the training of TCN.

There are several deep learning approaches to detect objects and their motion in videos. However, they require a large amount of annotated training data. Data annotation can be a very tedious process requiring lots of time and effort. In addition to that, training an object detection model can take up to a couple of days depending on the amount of training data and compute hardware. They can often end up being overkill, which restricts us from using deep learning approaches for object detection.

All the videos in our dataset had a camera operator off-frame bringing in a blue-colored sheet of paper into the frame and back out of it as shown in Figure 15(a) to signal to the demonstrator to start performing the task much like a clapperboard is used in film making to assist in picture and sound synchronization. Thus, we have a unique moving object whose direction of motion can be tracked and potentially be used to identify temporally similar video frames in out-of-sync multi-perspective videos. The object to be tracked is unique compared

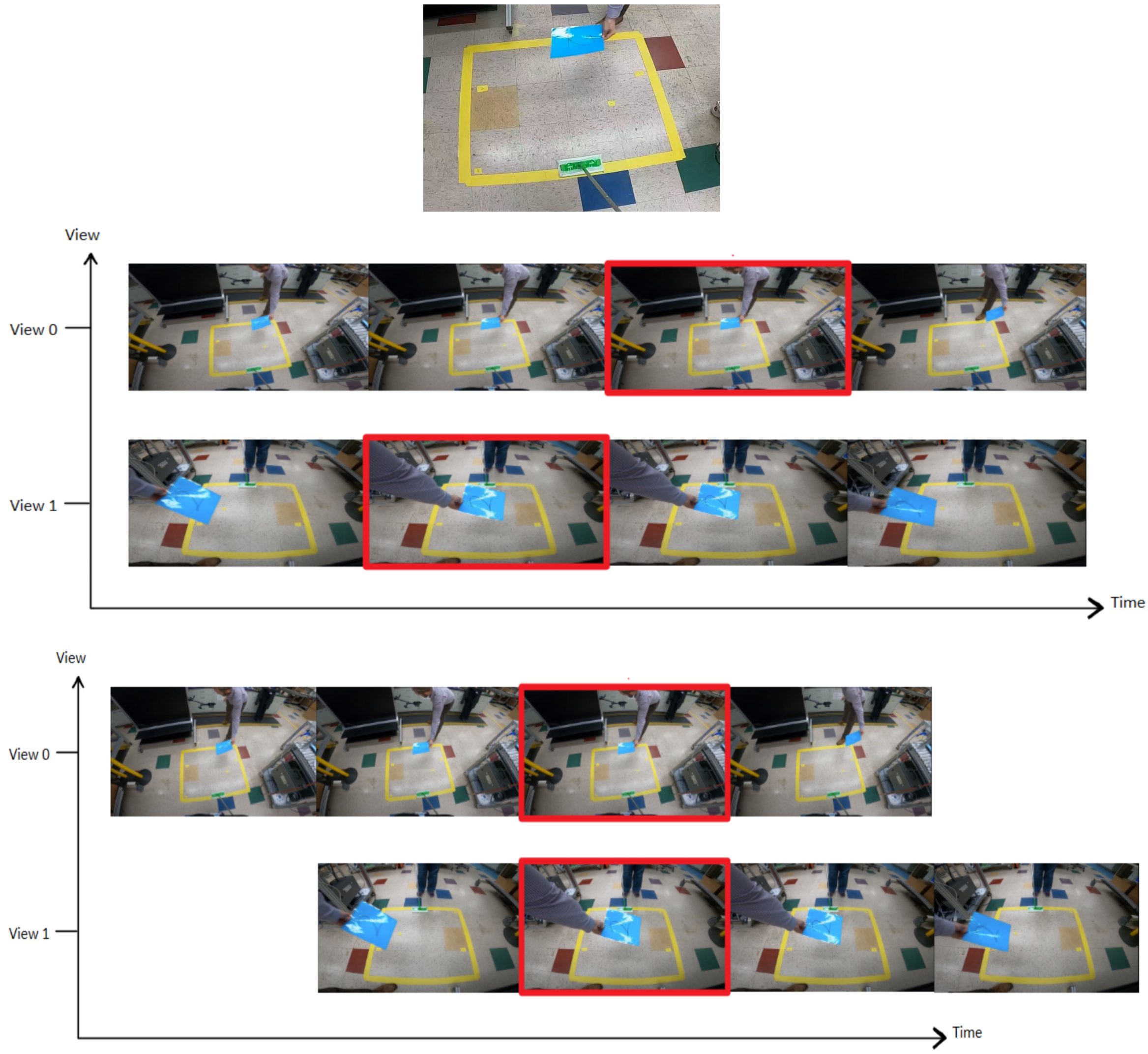


Figure 15. (a) Blue sheet to signal the demonstrator to start performing task (b) Temporal arrangement of frames before synchronization (c) Temporal arrangement of frames after synchronization.

to other objects in the video, so the object can be easily detected using a traditional computer vision approach.

Once the object is detected, its position and center in the image can easily be inferred. We do this for every video frame. When the object is in motion, before it changes direction, the 'x' and 'y' coordinates of its center are strictly increasing or decreasing. We keep on discarding the frames till this trend breaks. This event may occur at different points on each video of a different perspective. The resulting videos are perfectly synchronized because all frames were dropped before the occurrence of this event. Since the demonstration is not started until this event occurs, the dropped frames don't hold any significant information.

Figure 15(b) shows the temporal position of each frame of third-person and first-person videos of the same demonstration before alignment and Figure 15(c) shows their temporal positions after alignment. The highlighted frames in both the figures are the ones where the blue sheet changes direction.

Once the videos were satisfactorily aligned, they were clipped to segregate four different tasks. In total, there were 30 videos for each type of motion. Later, they were renamed to comply with requirements of the script used to convert them to TensorFlow record files having ".tfrecord" extension. This data format conversion helps TensorFlow process data faster and thus speeding up the training process.

4.3.4 Training TCN model

The input to the TCN is a batch of video frames (both first and third perspective). These video frames fall into three categories - anchor, positive and negative. Anchor frames are third person video frames that serve as reference, positive frames are first person frames that correspond to the same timestep as anchor frames and negative frames are first person video frames that are from a different timestep compared to anchor frames. The TCN then produces embeddings for each frame in the batch. To evaluate the outputs, we use triplet loss that specifies how far away the positive embeddings are from the anchor embeddings compared to negative embeddings. This loss is propagated backwards to train the model.

During training, frames in multiple views at the same time step are pushed close to each other in an "embedding" space. During the learning process, first person frame of the robot is continuously compared with the corresponding third person frame from an expert using the trained model. Closeness of these frames in the embedding space is obtained which can be used to let the robot know how closely it is imitating the expert.

4.3.5 Simulation of UR10e with mop

We simulated the UR10e robot holding a mop using ROS Kinetic and Gazebo. ROS provides a flexible control framework to calculate robot kinematics, joint trajectories and pass sensor messages. The UR10e robot is configured in ROS using the Universal Robots ROS Driver package which contains all of the base level robot data required for simulation and control. We created a mop end effector URDF (Universal Robot Description Format) that consists of a mop handle link connected to a mop head link by 2 degree of freedom revolute joint. The mop handle is attached to the tool flange of the robot by a fixed virtual joint. We simulated the robot with mop in Gazebo using the existing ROS interface. To create the Gazebo simulation world, we affix the robot base joint to a fixed table matching the height of the physical robot stand and added a yellow rectangle outline on the ground in the same location as the human demonstration video.

To control the robot Gazebo simulation, we used both direct joint angles and Cartesian planning using inverse kinematics. To start the simulation, the robot is first sent to a fixed home position using constant joint angles. The robot then follows a series of Cartesian points in X,Y,Z space to mimic the human demonstration video. The input waypoints calculated from the human demonstration video correspond to the position (x, y) of the mop head and not the robot. To solve for this, we use MoveIt and the URDF of the robot holding a mop to calculate the robot joint angles that correspond to a mop head pose using inverse kinematics.

4.3.6 Learning Human mopping to the robotic arm using RL

Due to time constraint we only learned the scrub task on the robot using RL. We selected a third-person human scrubbing video from test set and passed it to trained scrub TCN model and obtained the 32-bit embedding per frame for the video. Then we mount a wifi camera next to the robotic arm to record first person video of the robot performing the task. We use trajectory based RL optimization for learning the policy. So, we perform the mopping task with the current policy and record the trajectory video and update the policy at the end of each iteration. We start the training with 10 randomly generated samples and run 15 iterations to finally reach a policy which looks very similar to human mopping demonstrations. During training, the mop goes beyond the ground surface, so we had to perform most of the training in simulation until the robot is mopping on the ground surface and not mopping into the ground. Then we transfer the learned policy in simulation to a real robot and performed the final training of the mopping task on the robotic arm.

4.3.7 Evaluation of the Robot mopping system

First we evaluate our TCN model by passing two test videos from two different demonstrators (first-person video of one person and third-person video of another demonstrator) and observe how well the videos are paired. The pairing video examples can be seen in the attached video with the paper.

We use two methods to evaluate our mopping robotic system. The robotic arm perform the mopping task using a learned policy and we record the third-person view video for evaluation.

4.3.7.1 Cosine Similarity

We compared the third-person recorded video against the human third-person video demonstration to evaluate the robot imitation of human behavior from video. There are various techniques to measure the similarity between different images (116; 117), but since our experiments involve imitating behavior in different environments, we choose optical flow techniques to compare the motion between frames instead. Optical flow provides an estimation of motion based on changes in pixel intensity across time.

For cosine similarity [1] based evaluation, we use dense optical flow (finding the flow vectors for the entire frame), specifically the Farneback method (118) provided in OpenCV. Since the mop is constrained inside the yellow square, we first crop out the portions outside of the square

---

[1] https://en.wikipedia.org/wiki/Cosine_similarity

in all the frames of both the human and robot videos to reduce the effect of noise due to background movement. We then compute a mean vector from all the flow vectors obtained which represents the overall motion inside the yellow square on the ground, and compute a cosine similarity measure between the mean vector of the human frame and the mean vector of the robot frame. A similarity of +1 indicates that the mop is moving in the same direction in both the videos, whereas a similarity of -1 means that the mop is moving in opposite direction.

4.3.7.2 Euclidean Distance Loss

From our first method, we have obtained an approximate location of the mop in the human demonstration. We can compute the mop location while the robot is performing a task using learning RL policy. So, we can compute the Euclidean distance [1] between human demonstration mop location and the robot mop location.

4.4 Result

We achieved correct TCN pairing results from two different human demonstration videos as shown in the attached video. It indicates that our TCN model is trained properly. After training our robotic arm to mimic a human scrub motion (third-person video) using RL with our proposed reward function, we can see in the attached video that both methods successfully mimic the human mopping motion from video. The video shows that method 1 (TCN & RL) imitates the mopping motion better than method 2(Tracking & IK).

---

[1]https://en.wikipedia.org/wiki/Euclidean_distance

TABLE VII

EVALUATION RESULT OF THE TWO METHODS

	Average Cosine Similarity	Euclidean Distance Loss
Method 1 (Tracking and IK)	0.4752	0.126 m
Method 2 (TCN and RL)	0.7218	0.019 m

Table VII summarizes our evaluation of the third-person robot mopping using trained RL policy video versus third-person human mopping demonstration video. Column 2 in Table Table VII shows the cosine similarity comparison and column 3 shows the Euclidean distance comparison between two methods. The cosine similarity of method 1 is higher than method 2 which indicates that the mop is closely imitated using our method 2 than method 1. The Euclidean distance between the human mop and robot mop is very low for method 2 than 1. Thus, our proposed method 2 is successfully and accurately able to imitate the human mopping behavior from video.

4.5 Related Work

There are several efforts to build a robot system that can clean a surface by wiping or sweeping or scrubbing. Broadly, those can be divided into control-based methods and learning-based methods. For control-based methods, researchers use position-control methods (82; 83; 84; 85; 86; 87; 88; 89) for planning the trajectories for cleaning the surface using inverse kinematics (IK). For our first robotic system (baseline method), we took inspiration from position-control-based methods to move the robotic arm using IK once we obtain the position of the mop from the video. Several methods (90; 91; 92; 93; 94; 95) use force-control for constrained motion to produce cleaning actions. They use a force sensor attached to the robot feedback to plan the cleaning trajectory. However, force control fails to plan for continuously chaining force scenarios (uneven surfaces) and has some erroneous measurement from the force sensor. We plan to include force in our proposed learning-based robotic system as part of future work.

In the learning-based method, previous research (96; 97; 98) uses kinesthetic teaching to provide the robot with a motion to perform the sweeping task. To make the robot more autonomous for a cleaning task, (95; 99; 100; 101; 102; 103) developed a learning from demonstration approach using dynamic movement primitives (DMPs), Gaussian mixture model (GMM), Gaussian mixture regression (GMR), Hidden Markov model (HMM).

Imitation learning is a way to teach an agent (robot in our case) to perform a task by demonstration. Imitation learning has become a very popular method for teaching robots new skills or daily life activities (74; 75; 76; 77; 78; 79; 4). Recently, there is growing interest in learning skills from video demonstration (4; 79; 104; 119; 120; 121; 122; 123; 124). Sermanet et al. 2017 (4) proposed a Time Contrastive Network (TCN) to learn an embedding for the multi-view task performing video demonstration. They successfully performed a pouring task using a human pouring video demonstration on a Kuka robotic arm. For our proposed learning-based robotic system, we use the TCN approach to transform video mopping demonstration (high-level representation) to an embedding (low-level representation) which is used as a reward for the robot to mop like a human demonstration. We trained the TCN model using human mopping demonstration.

4.6 Conclusion and Future Work

In this chapter, we successfully built two robotic systems to mop like a human by learning from video demonstrations. First, we collected human mopping video demonstrations (first and third-person) for four different types of mopping motions. Second, we trained the TCN network for mopping motions and evaluated the performance of the trained TCN network. Then we devised a reward function for the mopping task and used RL to train the robotic arm to mimic human mopping behavior from the video demonstration. Both methods successfully mimic human mopping behavior. The learning-based method of TCN and RL imitates human behavior more precisely than the method on tracking and IK.

This research work opens up avenues to teach robots difficult daily tasks using video demonstration. We can use TCN plus other task-specific reward functions to teach robots daily household tasks. In this chapter, we build a robotic system with a fixed base. In the future, we can attach a mobile base to the robot and use the robot to move around and mop the floor. In this chapter, we imitate the position (x, y, and fixed z) of the mop from human mopping videos. However, for a mopping task, force is an important component for cleaning the floor. In future, we can incorporate force applied by human on the mop in our proposed robotic system. We can collect additional force data from human mopping by placing a load sensor on the mop. Then we can replace fixed z in our reward function with the corresponding force applied across the z-axis. Additionally we can use hybrid position-force control (125) methods to incorporate force in the mopping robotic system.

4.7 Acknowledgment

We thank Noah Kirst, Matthew Bauer and Su Chang for their help with data collection and system building. This research is supported By Procter & Gamble and P&G gift grant #2016-158710.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this Ph.D. thesis, we successfully solved three problems related to prediction and learning methods for collaborative robots.

We have successfully trained correspondence models for HTC Vive Controller to Baxter's arms. Our proposed deep model achieves better results than linear and non-linear regression baseline models for correspondence-based evaluations. In the real-time experiment, our deep network performed better than baselines model, resulting in faster completed tasks.

Further, we have proposed training inverse reinforcement learning models that were initially designed for policy estimation, to instead be optimized for goal prediction. We derived the gradient for optimizing goal likelihoods under the general discrete maximum entropy inverse reinforcement learning (MaxEnt IRL) setting and under the continuous inverse linear-quadratic regulation (LQR) setting. We demonstrated that our goal likelihood maximization method provides significant improvements for goal prediction compared to previous methods based on trajectory likelihood maximization in practice. Thus, with our new approach, we can more accurately infer intended goals farther in advance than previous approaches, enabling robots to know human intentions to make more compatible decisions.

Lastly, we solve a tedious daily life task, mopping the floor, where we perform imitation learning from human demonstrated videos of mopping the floor onto a robotic arm. We successfully built two robotic systems for learning mopping task from videos, one based on tracking and invese kinematics, and the second comprises of TCN and reinforcement learning. The robotic arm is able to mop similar to human videos using both methods. However, robot performs mopping more accurately using method 2 (TCN and RL) than the method 1 (tracking and inverse kinematics).

The work performed in this Thesis assumes that the robot space does not contain any obstacles. However, in real-life scenarios, we encounter obstacles most of the time. We can extend our methods by employing obstacles in the robot workspace.

In the future, we can combine Chapters two and three for building a robust robotic teleoperation system to enable robots to perform more complex tasks like peg hole insertion tasks. The human intent prediction method developed in Chapter three can predict human behaviors that can help in better planning of the robotic mopping motions for the Chapter four robot system.

For Chapter four mopping robotic system, we can use virtual reality-based robotic teleoperation methods from Chapter two to aid the RL training. The human in virtual reality can observe robot mopping training and correct them in real-time by providing correction feedback through VR controllers. It will reduce the training time and help the robotic arm learn accurate human mopping behaviors.

Lastly, we can extend Chapter four imitation learning for mopping tasks on a robotic arm to protocol-less videos or publicly available video sources example YouTube.

APPENDICES

Appendix A

IEEE COPYRIGHT POLICY

The IEEE does not require individuals working on a thesis to obtain a formal reuse license.

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis: 1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © [year of original publication] IEEE. 2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table 3) If you expect to use a substantial portion of the original paper, and if you are not the senior author, please obtain the senior author's approval. Requirements to be followed when using an entire IEEE copyrighted paper in a thesis: 1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication] 2) The published version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line. 3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted.

Appendix A (Continued)

Much of the work presented in chapter 2 has been accepted for publication in the proceedings of IEEE International Conference on Humanoid Humanoid Robots (ICHR) 2019 (24). Specific figures used in this thesis that are included in the accepted paper (24) are:

- Figure 4
- Figure 5
- Figure 6
- Figure 7
- Table I
- Table II
- Table III
- Table IV

please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

CITED LITERATURE

1. Gaurav, S. and Ziebart, B.: Discriminatively learning inverse optimal control models for predicting human intentions. In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, pages 1368–1376, 2019.

2. Jaynes, E. T.: Information theory and statistical mechanics. Physical Review, 106:620–630, 1957.

3. Schultz, C., Gaurav, S., Monfort, M., Zhang, L., and Ziebart, B. D.: Goal-predictive robotic teleoperation from noisy sensors. In IEEE International Conference on Robotics and Automation (ICRA), pages 5377–5383. IEEE, 2017.

4. Sermanet, P., Lynch, C., Chebotar, Y., Hsu, J., Jang, E., Schaal, S., and Levine, S.: Time-contrastive networks: Self-supervised learning from video. Proceedings of International Conference in Robotics and Automation (ICRA), 2018.

5. Abi-Farrajl, F., Henze, B., Werner, A., Panzirsch, M., Ott, C., and Roa, M. A.: Humanoid teleoperation using task-relevant haptic feedback. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 5010–5017, Oct 2018.

6. Ramos, J., Wang, A., Ubellacker, W., Mayo, J., and Kim, S.: A balance feedback interface for whole-body teleoperation of a humanoid robot and implementation in the hermes system. In Proceedings of IEEE-RAS 15th International Conference on Humanoid Robots, pages 844–850, Nov 2015.

7. Sian, N. E., Yokoi, K., Kajita, S., Kanehiro, F., and Tanie, K.: Whole body teleoperation of a humanoid robot - development of a simple master device using joysticks. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 2569–2574 vol.3, Sep. 2002.

8. Stilman, M., Koichi Nishiwaki, and Satoshi Kagami: Humanoid teleoperation for whole body manipulation. In Proceedings of IEEE International Conference on Robotics and Automation (ICRA), pages 3175–3180, May 2008.

9. Almetwally, I. and Mallem, M.: Real-time tele-operation and tele-walking of humanoid robot nao using kinect depth camera. In Proceedings of IEEE International Conference on Networking, Sensing and Control (ICNSC), pages 463–466, April 2013.

10. Vertut, J.: Teleoperation and robotics: applications and technology, volume 3. Springer Science & Business Media, 2013.

11. Zhang, T., McCarthy, Z., Jow, O., Lee, D., Goldberg, K., and Abbeel, P.: Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. arXiv preprint arXiv:1710.04615, 2017.

12. Theofilis, K., Orlosky, J., Nagai, Y., and Kiyokawa, K.: Panoramic view reconstruction for stereoscopic teleoperation of a humanoid robot. In Proceedings of IEEE International Conference on Humanoid Robots, pages 242–248, Nov 2016.

13. Fritsche, L., Unverzag, F., Peters, J., and Calandra, R.: First-person tele-operation of a humanoid robot. In Proceedings of IEEE-RAS International Conference on Humanoid Robots, pages 997–1002, Nov 2015.

14. Whitney, D., Rosen, E., Phillips, E., Konidaris, G., and Tellex, S.: Comparing Robot Grasping Teleoperation across Desktop and Virtual Reality with ROS Reality. In International Symposium on Robotics Research, 2017 in press.

15. Whitney, D., Rosen, E., Ullman, D., Phillips, E., and Tellex, S.: ROS Reality: A Virtual Reality Framework Using Consumer-Grade Hardware for ROS-Enabled Robots. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018.

16. Xi, B., Wang, S., Ye, X., Cai, Y., Lu, T., and Wang, R.: A robotic shared control teleoperation method based on learning from demonstrations. International Journal of Advanced Robotic Systems, 16(4):1729881419857428, 2019.

17. Suay, H. B. and Chernova, S.: Humanoid robot control using depth camera. In Proceedings of ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 401–401, March 2011.

18. Schultz, C.: Goal predictive infused robot teleoperation with kinect depth camera, Nov 2016.

19. Gaurav, S.: Goal-predictive robotic teleoperation using predictive filtering and goal change modeling, Apr 2017.

20. Sripada, A., Asokan, H., Warrier, A., Kapoor, A., Gaur, H., Patel, R., and R, S.: Teleoperation of a humanoid robot with motion imitation and legged locomotion. In 3rd International Conference on Advanced Robotics and Mechatronics (ICARM), pages 375–379, July 2018.

21. Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K.: Maximum entropy inverse reinforcement learning. In AAAI, pages 1433–1438, 2008.

22. Monfort, M., Liu, A., and Ziebart, B. D.: Intent prediction and trajectory forecasting via predictive inverse linear-quadratic regulation. In Proceedings of The Twenty-Ninth AAAI Conference on Artificial Intelligence, volume 5, pages 3672–3678, June 2015.

23. Koppula, H. S., Gupta, R., and Saxena, A.: Learning human activities and object affordances from rgb-d videos. The International Journal of Robotics Research, 32(8):951–970, 2013.

24. Gaurav, S., Al-Qurashi, Z., Barapatre, A., Maratos, G., Sarma, T., and Ziebart, B. D.: Deep correspondence learning for effective robotic teleoperation using virtual reality. In 2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids), pages 477–483, 2019.

25. Sokho Chang, Jungtae Kim, Insup Kim, Jin Hwan Borm, Chongwon Lee, and Jong Oh Park: Kist teleoperation system for humanoid robot. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)., volume 2, pages 1198–1203 vol.2, Oct 1999.

26. Sooyong Lee, Dae-Seong Choi, Munsang Kim, Chong-Won Lee, and Jae-Bok Song: An unified approach to teleoperation: human and robot integration. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)., pages 261–266 vol.1, Oct 1998.

27. Kofman, J., Xianghai Wu, Luu, T. J., and Verma, S.: Teleoperation of a robot manipulator using a vision-based human-robot interface. IEEE Transactions on Industrial Electronics, 52(5):1206–1219, Oct 2005.

28. Rodriguez, I., Astigarraga, A., Jauregi, E., Ruiz, T., and Lazkano, E.: Humanizing nao robot teleoperation using ros. In Proceedings of IEEE International Conference on Humanoid Robots, pages 179–186, Nov 2014.

29. Borges, M., Symington, A., Coltin, B., Smith, T., and Ventura, R.: Htc vive: Analysis and accuracy improvement. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 2610–2615. IEEE, 2018.

30. Ju, Z., Yang, C., and Ma, H.: Kinematics modeling and experimental verification of baxter robot. In Chinese Control Conference (CCC), pages 8518–8523. IEEE, 2014.

31. Goodfellow, I. J., Bengio, Y., and Courville, A. C.: Deep Learning. Adaptive computation and machine learning. MIT Press, 2016.

32. Psaltis, D., Sideris, A., and Yamamura, A. A.: A multilayered neural network controller. IEEE control systems magazine, 8(2):17–21, 1988.

33. Bekey, G. A. and Goldberg, K. Y.: Neural Networks in robotics, volume 202. Springer Science & Business Media, 2012.

34. Nielsen, M. A.: Neural networks and deep learning. Determination Press, 2015.

35. Demuth, H. and Beale, M.: Neural network toolbox. For Use with MATLAB. The MathWorks Inc, 2000, 1992.

36. Goldman, R.: Understanding quaternions. Graphical models, 73(2):21–49, 2011.

37. Ben-Ari, M.: A tutorial on euler angles and quaternions. Weizmann Institute of Science, Israel, 2014.

38. Kingma, D. P. and Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

39. Argall, B. D., Chernova, S., Veloso, M., and Browning, B.: A survey of robot learning from demonstration. Robotics and autonomous systems, 57(5):469–483, 2009.

40. Strabala, K., Lee, M. K., Dragan, A., Forlizzi, J., Srinivasa, S., Cakmak, M., and Micelli, V.: Towards seamless human-robot handovers. Journal of Human-Robot Interaction, January 2013.

41. Rozo, L., Amor, H. B., Calinon, S., Dragan, A., and Lee, D.: Special issue on learning for human–robot collaboration. Autonomous Robots, 42(5):953–956, Jun 2018.

42. Reddy, S., Dragan, A., and Levine, S.: Shared autonomy via deep reinforcement learning. In Proceedings of Robotics: Science and Systems, Pittsburgh, Pennsylvania, June 2018.

43. Demiris, Y.: Prediction of intent in robotics and multi-agent systems. Cognitive Processing, 8(3):151–158, Sep 2007.

44. Pan, X. and Shen, Y.: Human-interactive subgoal supervision for efficient inverse reinforcement learning. In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '18, pages 1380–1387, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

45. Trafton, J. G., Hiatt, L. M., Harrison, A. M., Tamborello, II, F. P., Khemlani, S. S., and Schultz, A. C.: Act-r/e: An embodied cognitive architecture for human-robot interaction. J. Hum.-Robot Interact., 2(1):30–55, February 2013.

46. Lee, J.: A survey of robot learning from demonstrations for human-robot collaboration. arXiv preprint arXiv:1710.08789, 2017.

47. Bajcsy, A., Losey, D. P., O'Malley, M. K., and Dragan, A. D.: Learning robot objectives from physical human interaction. In Proceedings of the 1st Annual Conference on Robot Learning, eds. S. Levine, V. Vanhoucke, and K. Goldberg, volume 78 of Proceedings of Machine Learning Research, pages 217–226. PMLR, 13–15 Nov 2017.

48. Banerjee, S. and Chernova, S.: Temporal models for robot classification of human interruptibility. In Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS '17, pages 1350–1359, Richland, SC, 2017. International Foundation for Autonomous Agents and Multiagent Systems.

49. Muelling, K., Venkatraman, A., Valois, J.-S., Downey, J., Weiss, J., Javdani, S., Hebert, M., Schwartz, A. B., Collinger, J. L., and Bagnell, J. A. D.: Autonomy infused teleoperation with application to bci manipulation. In Proceedings of Robotics: Science and Systems, July 2015.

50. Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K.: Human behavior modeling with maximum entropy inverse optimal control. In Association for the Advancement of Artificial Intelligence Spring Symposium: Human Behavior Modeling, 2009.

51. Dragan, A. D., Lee, K. T., and Srinivasa, S. S.: Legibility and predictability of robot motion. In Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction, pages 301–308, 2013.

52. Baker, C. L., Tenenbaum, J. B., and Saxe, R. R.: Goal inference as inverse planning. In Proceedings of the Annual Meeting of the Cognitive Science Society, volume 29, 2007.

53. Ghahramani, Z.: An introduction to hidden markov models and bayesian networks. International journal of pattern recognition and artificial intelligence, 15(01):9–42, 2001.

54. Simmons, R., Browning, B., Zhang, Y., and Sadekar, V.: Learning to predict driver route and destination intent. In 2006 IEEE Intelligent Transportation Systems Conference, pages 127–132, Sept 2006.

55. Krumm, J. and Horvitz, E.: Predestination: Inferring destinations from partial trajectories. In Proc. Ubicomp, pages 243–260, 2006.

56. Marmasse, N. and Schmandt, C.: A user-centered location model. Personal and Ubiquitous Computing, 6(5):318–321, Dec 2002.

57. Holladay, R. M., Dragan, A. D., and Srinivasa, S. S.: Legible robot pointing. In Robot and Human Interactive Communication, 2014 RO-MAN: The 23rd IEEE International Symposium on, pages 217–223. IEEE, 2014.

58. Ziebart, B. D.: Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy. Doctoral dissertation, Carnegie Mellon University, 2010.

59. Takagi, R., Kitamura, Y., Naito, S., and Kishino, F.: A fundamental study on error-corrective feedback movement in a positioning task. In Proc. of Asian Pacific Computer Human Interaction, pages 160–172, 2002.

60. Asano, T., Sharlin, E., Kitamura, Y., Takashima, K., and Kishino, F.: Predictive interaction using the delphian desktop. In Proceedings of the Annual ACM symposium

on User Interface Software and Technology, page 141. ACM, Proceedings of the Annual ACM symposium on User Interface Software and Technology, 2005.

61. James, G., Witten, D., Hastie, T., and Tibshirani, R.: An introduction to statistical learning, volume 6. Springer, 2013.

62. Chaudhary, A.: Discriminative predictive analysis for goal prediction, Dec 2017.

63. Koppula, H. S. and Saxena, A.: Anticipating human activities using object affordances for reactive robotic response. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(1):14–29, Jan 2016.

64. Kalman, R.: When is a linear control system optimal? Trans. ASME, J. Basic Engrg., 86:51–60, 1964.

65. Ng, A. Y. and Russell, S. J.: Algorithms for inverse reinforcement learning. In Proceedings of the Seventeenth International Conference on Machine Learning, ICML '00, pages 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

66. Abbeel, P. and Ng, A. Y.: Apprenticeship learning via inverse reinforcement learning. In Proceedings of the Twenty-first International Conference on Machine Learning, ICML '04, pages 1–, New York, NY, USA, 2004. ACM.

67. Jaynes, E. T.: Information theory and statistical mechanics, II. Physical review, 108(2):171–190, 1957.

68. Ziebart, B. D., Bagnell, J. A., and Dey, A. K.: The principle of maximum causal entropy for estimating interacting processes. IEEE Transactions on Information Theory, 59(4):1966–1980, 2013.

69. Duchi, J., Hazan, E., and Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res., 12:2121–2159, July 2011.

70. Sutskever, I., Martens, J., Dahl, G., and Hinton, G.: On the importance of initialization and momentum in deep learning. In Proceedings of the 30th International Conference on Machine Learning, eds. S. Dasgupta and D. McAllester, volume 28 of Proceedings of Machine Learning Research, pages 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

71. Sanderson, C.: Armadillo: An open source c++ linear algebra library for fast prototyping and computationally intensive experiments, 2010.

72. Spong, M. W. and Vidyasagar, M.: Robot dynamics and control. John Wiley & Sons, 2008.

73. Byravan, A., Montfort, M., Ziebart, B., Boots, B., and Fox, D.: Layered hybrid inverse optimal control for learning robot manipulation from demonstration. In NIPS workshop on autonomous learning robots. Citeseer, 2014.

74. Fang, B., Jia, S., Guo, D., Xu, M., Wen, S., and Sun, F.: Survey of imitation learning for robotic manipulation. International Journal of Intelligent Robotics and Applications, 3(4):362–369, 2019.

75. Osa, T., Pajarinen, J., Neumann, G., Bagnell, J. A., Abbeel, P., and Peters, J.: An algorithmic perspective on imitation learning. arXiv preprint arXiv:1811.06711, 2018.

76. Zhu, Y., Wang, Z., Merel, J., Rusu, A., Erez, T., Cabi, S., Tunyasuvunakool, S., Kramár, J., Hadsell, R., de Freitas, N., et al.: Reinforcement and imitation learning for diverse visuomotor skills. arXiv preprint arXiv:1802.09564, 2018.

77. Zhang, T., McCarthy, Z., Jow, O., Lee, D., Chen, X., Goldberg, K., and Abbeel, P.: Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 5628–5635. IEEE, 2018.

78. Hussein, A., Gaber, M. M., Elyan, E., and Jayne, C.: Imitation learning: A survey of learning methods. ACM Computing Surveys (CSUR), 50(2):1–35, 2017.

79. Peng, X. B., Kanazawa, A., Malik, J., Abbeel, P., and Levine, S.: Sfv: Reinforcement learning of physical skills from videos. ACM Transactions On Graphics (TOG), 37(6):1–14, 2018.

80. Chebotar, Y., Hausman, K., Zhang, M., Sukhatme, G., Schaal, S., and Levine, S.: Combining model-based and model-free updates for trajectory-centric reinforcement learning. In Proceedings of the 34th International Conference on Machine Learning, eds. D. Precup and Y. W. Teh, volume 70 of Proceedings of Machine Learning Research, pages 703–711. PMLR, 06–11 Aug 2017.

81. Kim, J., Mishra, A. K., Limosani, R., Scafuro, M., Cauli, N., Santos-Victor, J., Mazzolai, B., and Cavallo, F.: Control strategies for cleaning robots in domestic applications: A comprehensive review. International Journal of Advanced Robotic Systems, 16(4):1729881419857432, 2019.

82. Hess, J., Tipaldi, G. D., and Burgard, W.: Null space optimization for effective coverage of 3d surfaces using redundant manipulators. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1923–1928. IEEE, 2012.

83. Bormann, R., Hampp, J., and Hägele, M.: New brooms sweep clean-an autonomous robotic cleaning assistant for professional office cleaning. In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 4470–4477. IEEE, 2015.

84. King, C.-H., Chen, T. L., Jain, A., and Kemp, C. C.: Towards an assistive robot that autonomously performs bed baths for patient hygiene. In 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 319–324. IEEE, 2010.

85. Okada, K., Kojima, M., Sagawa, Y., Ichino, T., Sato, K., and Inaba, M.: Vision based behavior verification system of humanoid robot for daily environment tasks. In 2006 6th IEEE-RAS International Conference on Humanoid Robots, pages 7–12. IEEE, 2006.

86. Liang, J., Zhang, G., Wang, W., Hou, Z., Li, J., Wang, X., and Han, C.-S.: Dual quaternion based kinematic control for yumi dual arm robot. In 2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), pages 114–118. IEEE, 2017.

87. Yamazaki, K., Ueda, R., Nozawa, S., Mori, Y., Maki, T., Hatao, N., Okada, K., and Inaba, M.: System integration of a daily assistive robot and its application to tidying and cleaning rooms. In 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1365–1371. IEEE, 2010.

88. Sato, F., Nishii, T., Takahashi, J., Yoshida, Y., Mitsuhashi, M., and Nenchev, D.: Experimental evaluation of a trajectory/force tracking controller for a humanoid robot cleaning a vertical surface. In 2011 IEEE/RSJ international conference on intelligent robots and systems, pages 3179–3184. IEEE, 2011.

89. Lana, E. P., Adorno, B. V., and Maia, C. A.: A new algebraic approach for the description of robotic manipulation tasks. In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 3083–3088. IEEE, 2015.

90. Ortenzi, V., Adjigble, M., Kuo, J. A., Stolkin, R., and Mistry, M.: An experimental study of robot control during environmental contacts based on projected operational space dynamics. In 2014 IEEE-RAS International Conference on Humanoid Robots, pages 407–412. IEEE, 2014.

91. Leidner, D. and Beetz, M.: Inferring the effects of wiping motions based on haptic perception. In 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids), pages 461–468. IEEE, 2016.

92. Rocchi, A., Hoffman, E. M., Caldwell, D. G., and Tsagarakis, N. G.: Opensot: a whole-body control library for the compliant humanoid robot coman. In 2015 IEEE International Conference on Robotics and Automation (ICRA), pages 6248–6253. IEEE, 2015.

93. Leidner, D., Dietrich, A., Beetz, M., and Albu-Schäffer, A.: Knowledge-enabled parameterization of whole-body control strategies for compliant service robots. Autonomous Robots, 40(3):519–536, 2016.

94. Leidner, D., Bejjani, W., Albu-Schäffer, A., and Beetz, M.: Robotic agents representing, reasoning, and executing wiping tasks for daily household chores. AUTONOMOUS AGENTS AND MULTI-AGENT SYSTEMS, 2016.

95. Urbanek, H., Albu-Schaffer, A., and van der Smagt, P.: Learning from demonstration: repetitive movements for autonomous service robotics. In 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), volume 4, pages 3495–3500. IEEE, 2004.

96. Eppner, C., Sturm, J., Bennewitz, M., Stachniss, C., and Burgard, W.: Imitation learning with generalized task descriptions. In 2009 IEEE International Conference on Robotics and Automation, pages 3968–3974. IEEE, 2009.

97. Attamimi, M., Araki, T., Nakamura, T., and Nagai, T.: Visual recognition system for cleaning tasks by humanoid robots. International Journal of Advanced Robotic Systems, 10(11):384, 2013.

98. Kabir, A. M., Langsfeld, J. D., Zhuang, C., Kaipa, K. N., and Gupta, S. K.: Automated learning of operation parameters for robotic cleaning by mechanical scrubbing. In ASME 2016 11th International Manufacturing Science and Engineering Conference. American Society of Mechanical Engineers Digital Collection, 2016.

99. Gams, A. and Ude, A.: On-line coaching of robots through visual and physical interaction: Analysis of effectiveness of human-robot interaction strategies. In 2016 IEEE International Conference on Robotics and Automation (ICRA), pages 3028–3034. IEEE, 2016.

100. Elliott, S., Xu, Z., and Cakmak, M.: Learning generalizable surface cleaning actions from demonstration. In 2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), pages 993–999. IEEE, 2017.

101. Dometios, A. C., Zhou, Y., Papageorgiou, X. S., Tzafestas, C. S., and Asfour, T.: Vision-based online adaptation of motion primitives to dynamic surfaces: application to an interactive robotic wiping task. IEEE Robotics and Automation Letters, 3(3):1410–1417, 2018.

102. Kim, J., Cauli, N., Vicente, P., Damas, B., Cavallo, F., and Santos-Victor, J.: "icub, clean the table!" a robot learning from demonstration approach using deep neural networks. In 2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pages 3–9. IEEE, 2018.

103. Silvério, J., Rozo, L., Calinon, S., and Caldwell, D. G.: Learning bimanual end-effector poses from demonstrations using task-parameterized dynamical systems. In 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS), pages 464–470. IEEE, 2015.

104. Liu, Y., Gupta, A., Abbeel, P., and Levine, S.: Imitation from observation: Learning to imitate behaviors from raw video via context translation. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 1118–1125. IEEE, 2018.

105. Devin, C., Abbeel, P., Darrell, T., and Levine, S.: Deep object-centric representations for generalizable robot learning. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 7111–7118. IEEE, 2018.

106. Friedman, J., Hastie, T., Tibshirani, R., et al.: Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). Annals of

statistics, 28(2):337–407, 2000.

107. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A. Y., et al.: Ros: an open-source robot operating system. In ICRA workshop on open source software, volume 3, page 5. Kobe, Japan, 2009.

108. Liu, S. and Liu, P.: A review of motion planning algorithms for robotic arm systems. In 8th International Conference on Robot Intelligence Technology and Applications (proceedings). IEEE, 2020.

109. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z.: Rethinking the inception architecture for computer vision. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2818–2826, 2016.

110. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee, 2009.

111. Schroff, F., Kalenichenko, D., and Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 815–823, 2015.

112. Levine, S. and Abbeel, P.: Learning neural network policies with guided policy search under unknown dynamics. In NIPS, volume 27, pages 1071–1079. Citeseer, 2014.

113. Tassa, Y., Erez, T., and Todorov, E.: Synthesis and stabilization of complex behaviors through online trajectory optimization. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 4906–4913. IEEE, 2012.

114. Theodorou, E., Buchli, J., and Schaal, S.: A generalized path integral control approach to reinforcement learning. The Journal of Machine Learning Research, 11:3137–3181, 2010.

115. Chebotar, Y., Kalakrishnan, M., Yahya, A., Li, A., Schaal, S., and Levine, S.: Path integral guided policy search. In 2017 IEEE international conference on robotics and automation (ICRA), pages 3381–3388. IEEE, 2017.

116. Sinha, P. and Russell, R.: A perceptually based comparison of image similarity metrics. Perception, 40(11):1269–1281, 2011.

117. Kordopatis-Zilos, G., Papadopoulos, S., Patras, I., and Kompatsiaris, I.: Visil: Fine-grained spatio-temporal video similarity learning. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 6351–6360, 2019.

118. Farnebäck, G.: Two-frame motion estimation based on polynomial expansion. In Image Analysis, eds. J. Bigun and T. Gustavsson, pages 363–370, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

119. Torabi, F., Warnell, G., and Stone, P.: Imitation learning from video by leveraging proprioception. arXiv preprint arXiv:1905.09335, 2019.

120. Dwibedi, D., Tompson, J., Lynch, C., and Sermanet, P.: Learning actionable representations from visual observations. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1577–1584. IEEE, 2018.

121. Chen, A. S., Nair, S., and Finn, C.: Learning generalizable robotic reward functions from" in-the-wild" human videos. arXiv preprint arXiv:2103.16817, 2021.

122. Petrík, V., Tapaswi, M., Laptev, I., and Sivic, J.: Learning object manipulation skills via approximate state estimation from real videos. arXiv preprint arXiv:2011.06813, 2020.

123. Schmeckpeper, K., Rybkin, O., Daniilidis, K., Levine, S., and Finn, C.: Reinforcement learning with videos: Combining offline observations with interaction. arXiv preprint arXiv:2011.06507, 2020.

124. Das, N., Bechtle, S., Davchev, T., Jayaraman, D., Rai, A., and Meier, F.: Model-based inverse reinforcement learning from visual demonstrations. arXiv preprint arXiv:2010.09034, 2020.

125. Spong, M. W., Hutchinson, S., Vidyasagar, M., et al.: Robot modeling and control, volume 3. wiley New York, 2006.

126. Bishop, C.: Pattern recognition and machine learning (information science and statistics), 1st edn. 2006. corr. 2nd printing edn. Springer, New York, 2007.

127. Andersen, M. R., Jensen, T., Lisouski, P., Mortensen, A. K., Hansen, M. K., Gregersen, T., and Ahrendt, P.: Kinect depth sensor evaluation for computer vision applications. Electrical and Computer Engineering Technical Report ECE-TR-6, 2012.

128. Zhang, Z.: Microsoft kinect sensor and its effect. IEEE multimedia, 19(2):4–10, 2012.

129. Murphy, K. P.: Machine Learning: A Probabilistic Perspective. MIT press, 2012.

130. Sian, N. E., Yokoi, K., Kajita, S., and Tanie, K.: Whole body teleoperation of a humanoid robot integrating operator's intention and robot's autonomy: an experimental verification. In Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), pages 1651–1656 vol.2, Oct 2003.

131. Sian, N., Yokoi, K., Kajita, S., Kanehiro, F., and Tanie, K.: Whole body teleoperation of a humanoid robot - a method of integrating operator's intention and robot's autonomy. In IEEE International Conference on Robotics and Automation, pages 1613–1619 vol.2, Sep. 2003.

132. Quintero, C. P., Fomena, R. T., Shademan, A., Ramirez, O., and Jagersand, M.: Interactive teleoperation interface for semi-autonomous control of robot arms. In 2014 Canadian Conference on Computer and Robot Vision, pages 357–363, May 2014.

133. Field, M., Stirling, D., Naghdy, F., and Pan, Z.: Motion capture in robotics review. In 2009 IEEE International Conference on Control and Automation, pages 1697–1702. IEEE, 2009.

134. Finn, C., Abbeel, P., and Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML'17, page 1126–1135. JMLR.org, 2017.

135. Redmon, J., Divvala, S., Girshick, R., and Farhadi, A.: You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 779–788, 2016.

VITA

NAME: Sanket Gaurav

EDUCATION: B.Tech., Computer Science and Engineering, Sikkim Manipal Institute of Technology, 2014

M.S., Computer Science, University of Illinois at Chicago, 2017

Ph.D., Computer Science, University of Illinois at Chicago, 2021

EXPERIENCE: University of Illinois at Chicago, Chicago, IL, Graduate Teaching Assistant, Aug 2016- May 2019

Amazon, Seattle, WA Applied Scientist Intern, May 2020- Aug 2020

Procter & Gamble, Cincinnati, OH, Deep Reinforcement Learning Scientist Intern, May 2019- Dec 2019

Pacific Northwest National Laboratory, Masters Intern, June 2016- Aug 2016

PUBLICATION: Gaurav, S., and Ziebart, B. D.: Discriminatively Learning Inverse Optimal Control Models for Predicting Human Intentions. accepted for publication at ACM AAMAS 2019

Gaurav, S., Al-Qurashi, Z., Barapatre, A., Maratos, G., Sarma, T., and Ziebart, B. D.: Deep Correspondence Learning for Effective Robotic Teleoperation using Virtual Reality. accepted for publication at IEEE HUMANOIDS 2019