Goal Predictive Infused Robot Teleoperation with Kinect Depth Camera

by

Christopher Schultz
B.S., University of Wisconsin - Madison, 2011

Thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2016

Chicago, Illinois

Defense Committee:
Brian D. Ziebart, Chair and Advisor
Piotr J. Gmytrasiewicz
Milos Zefran, Electrical and Computer Engineering

This thesis is dedicated to Margaret Hemphill and my family. Without their unwavering support, this work would not have been possible.

ACKNOWLEDGMENT

I would like to thank my advisor - Professor Brian Ziebart - who oversaw the development of this work over the past two years. I would also like to thank Professor Piotr Gmytrasiewicz and Professor Milos Zefran for their participation on my Masters thesis committee.

I would like to thank my fellow co-authors of the submitted ICRA 2017 manuscript of this work: Sanket Gaurav, Mathew Monfort, and Lingfei Zhang. Key machine learning algorithms utilized in this work were developed in Mathew Monforts PhD work at UIC. Sanket Gaurav and Lingfei Zhang assisted greatly in this work by writing code and providing valuable opinions. See the appendices for details about this submitted manuscript.

I would like to thank the undergraduates who assisted me on this work throughout the past two years: David Labek, Filip Radzikowski, and Chris Griffith.

Finally, I would to thank the National Science Foundation (NSF) for its grant in support of this work specifically through NSF NRI Award No. 1227495.

CS

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BCI	Brain-Computer Interface
DOF	Degrees of Freedom
ICRA	IEEE International Conference on Robotics and Automation
IEEE	Institute of Electrical and Electronics Engineers
IR	Infrared Spectrum Light
LQR	Linear-Quadratic Control
RGB	Red, Green, and Blue video data

SUMMARY

Using depth camera technology developed in recent years, a human's pose demonstrations can be used as inputs into unilateral robotic teleoperation. This teleoperation system provides an intuitive and effective means of control for the human operator. However, the imprecision of low-cost depth cameras and difficulties with the frame of reference for the human operator introduce inefficiencies in the teleoperation process when performing tasks that require precise robotic interactions with the robot's work space.

We developed a goal-predictive teleoperation system that addresses these difficulties for the human operator by adding goal-directed aid to the teleoperation control process. Our approach used inverse optimal control to predict the intended final state of the robotic system from the current motion trajectory in real time and then adapted the degree of autonomy between the operator's demonstrations and autonomous completion of the predicted task. We evaluated our approach by using a Microsoft Kinect depth camera as an input sensor to control a Rethink Robotics Baxter robot. The results verify the effectiveness of our developed goal-predictive teleoperation system.

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

Effective robots are critically needed in a number of settings where human capabilities are limited. These include operation in settings that are unsafe for a human presence [1], when lifting heavy objects that exceed the limits of human physical strength [2], and/or when precise movements are needed at fine scales that are beyond the precision of unaided human motor control [3]. Despite significant advancements in artificial intelligence [4,5], human teleoperators are still more adept at many of the robotic motion planning and manipulation tasks [6] encountered in these settings, which require versatility and high-level problem solving. Furthermore, programming by demonstration through teleoperation is an attractive modality for enabling end-users without computer programming expertise to create desired autonomous behavior [7]. Methods that improve the efficiency of teleoperation by sharing autonomy between human operator and autonomous controller [8, 9] –leveraging the advantages of each– are needed to further improve the efficiency of completing these tasks.

A natural input mechanism for humanoid robot teleoperation is for the end-user to simply demonstrate the desired robot movements to a passively observing sensor and have the movements imitated by the robot [10–12]. There are two key challenges to this form of unilateral teleoperation. First, the translation from human to robotic poses is often complicated due to

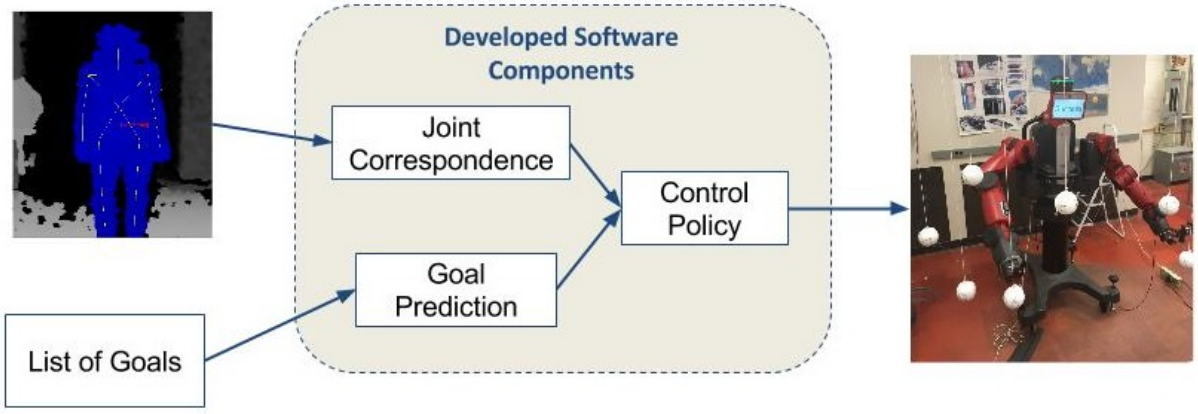

Figure 1. An overview of our developed teleoperation system.

differences in physical embodiment. Joint angle limitations of human operators and humanoid robots can differ significantly, for example. Second, apart from motion capture systems [13–15], which are very precise but require a calibrated environment and are relatively expensive, human pose estimates are often susceptible to sensor noise. Depth cameras, like the Microsoft Kinect, are relatively inexpensive, but can produce skeleton tracking errors that may make them unsuitable for fine-grained teleoperation tasks on their own.

1.2 Thesis Statement

We investigated goal-predictive pose-based robotic teleoperation from depth camera data: the task of using the knowledge of possible task completion goals to improve the efficiency of robotic teleoperation from depth camera data [8]. Our developed approach is composed of three main steps. First, we developed a correspondence between the human operator's tracked skeleton and robotic joint positions. This correspondence is used to translate from human

operator pose to a robot pose. Second, we learned a model for goal prediction using inverse optimal control for linear-quadratic systems [16]. This model provides a posterior probability estimate for each possible target or goal in the robot's work space given the partial trajectory. Third, we applied control assistance policies based on the confidence of the goal prediction component to robotic teleoperation. These policies increase the autonomy of the controller when predicted goal certainty is high. We evaluated our approach experimentally on teleoperated pointing tasks using the Baxter robot from Rethink Robotics [17] as the robotic platform and a Microsoft Kinect [18] as the input sensor.

1.3 Outline of Document

The order of discussion in this document is as follows:

First, we discuss the associated background technologies and relevant topics in the domains of robotics and machine learning in Chapter 2. For example, we review depth camera technology and the principles of inverse optimal control.

Next, we discuss the three main components of the our approach in separate chapters: (1) learning a correspondence between human operator pose and robot pose; (2) predicting the intended goal of the operator given a partial trajectory; and (3) control assistance when goal prediction confidence is high.

Then, we detail the experimental setup used to validate the developed goal-predictive teleoperation system in Chapter 6. We discuss the experimental results as well which validate the success of the developed goal-predictive teleoperation system.

Finally, in Chapter 7, we discuss the conclusions of this thesis and future extensions. There are many possible extensions of our developed system that can build on the success of this thesis.

CHAPTER 2

BACKGROUND

2.1 Depth Camera and OpenNI Skeleton Tracking

A depth camera gathers both depth and traditional RGB video data of its field of view. The type of depth camera used in this work is structured-light based, specifically Microsoft's Kinect camera. Structured-light depth cameras obtain depth data by projecting a structured pattern of light onto its field of view. In the case of the Kinect camera, the pattern projected is in the IR spectrum of light to not interfere with the visible light of the scene. A camera at a baseline distance b away from the projection source observes the projected pattern. Because the pattern detection camera has a different perspective on the scene than the projection source, the project pattern will appear distorted from the perspective of the camera [19]. This is referred to as the disparity of the pattern, notated as $m(x, y)$ and can be measured by a depth camera. This disparity can be used to calculate the depth of a pixel through [19]:

$$depth = \frac{b * f}{m(x, y)} \tag{2.1}$$

where the f is the focal length of the pattern detecting camera. With camera calibration, RGB data of the scene, gathered from a separate camera than the pattern detection camera, can be combined with the depth calculated from Equation 2.1 to create 3-d data. The Kinect camera can specifically gather 3-d data captions at 30 Hz at a resolution of 640 by 480 [19].

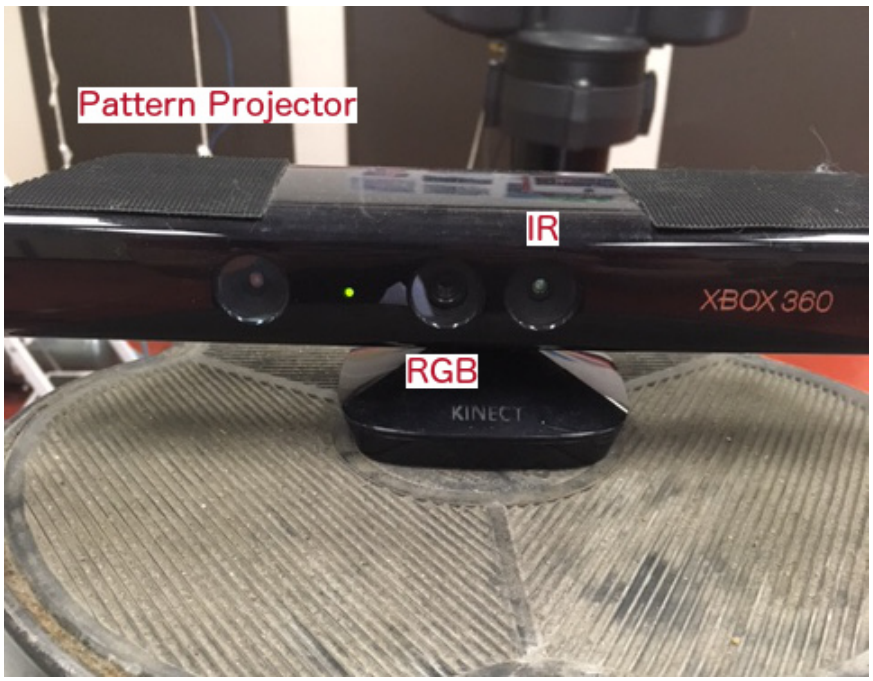

Figure 2. Layout of a Microsoft Kinect camera.

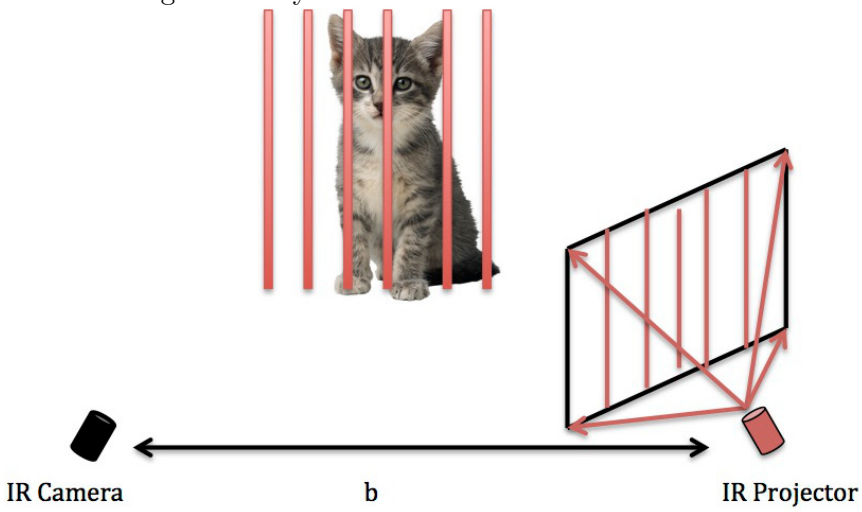

Figure 3. Example projected pattern setup used by structured-light type depth cameras.

The data outputted from a depth camera is point cloud formatted data. Point cloud data is a matrix of pixels with each pixel having x,y,z data and RGB color data. A visualized example of point cloud data can be seen in Figure 4. Computer vision algorithms have been developed to segment "humanoid" shapes out of raw point cloud data, [20] for example. The OpenNI software framework [1] leverages segmentation algorithms to extract "humanoid" shapes out of a point data cloud. The OpenNI framework then overlays human skeleton models on the segmented data. The skeleton model data consists of 15 skeleton points each having x,y,z translation and w,x,y,z rotational data (105 total data points). A visualized example of the OpenNI output can be seen in Figure 5.

Each "humanoid" shape in the depth camera's field of view is considered as a separate user by the OpenNI framework and tracked with a separate skeleton model. The OpenNI framework does require some calibration prior to the skeleton tracking of a user. However, the calibration is handled automatically when the OpenNI framework detects a new user in the associated depth camera's field of vision.

---

[1] `https://github.com/ros-drivers/openni_launch`

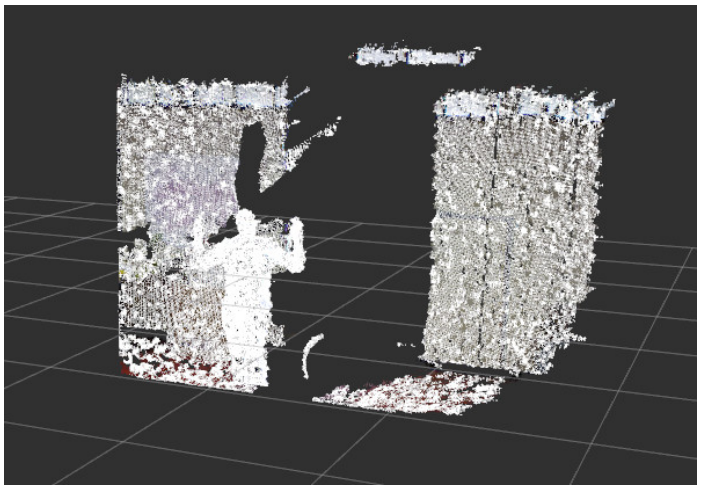

Figure 4. Visualized example of point cloud data.

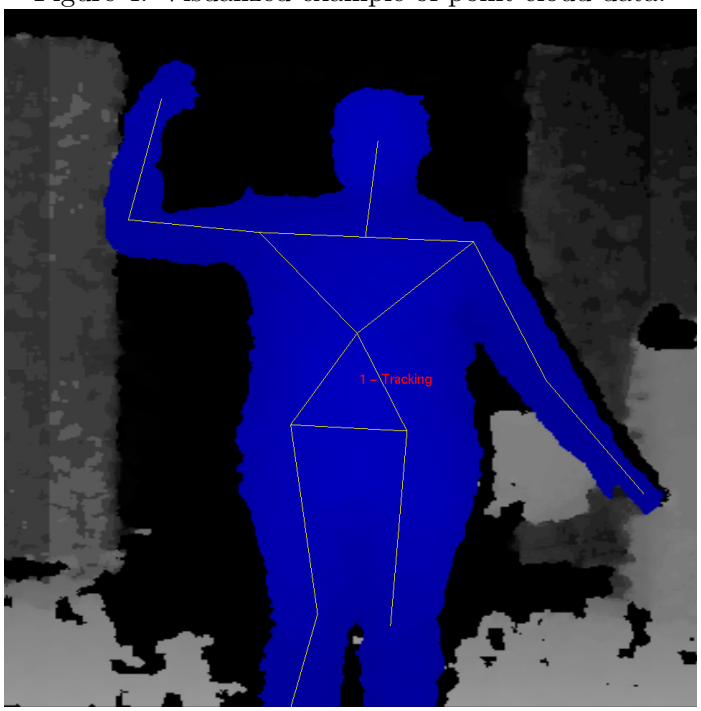

Figure 5. Visualized Example of OpenNI skeleton tracking data.

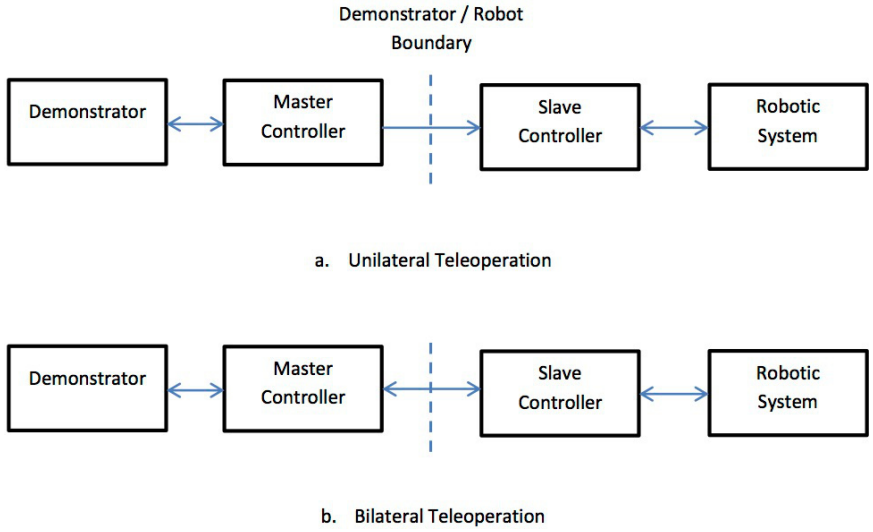

Figure 6. Control scheme differences between unilateral and bilateral teleoperation.

2.2 Robotic Teleoperation

Robotic teleoperation refers to robotic control with human participation [21]. There are generally two types of teleoperation systems: unilateral and bilateral control. In unilateral control, a human teleoperator provides control inputs to a robot through a master control interface, such as a joystick, which then provides input into a robot slave controller. The slave controller performs the robotic system manipulations based off of the output from the master control interface. Bilateral control is unilateral control with an additional feedback loop in which a robotic slave controller provides feedback to the master control interface. This additional feedback allows the master control interface to provide feedback to human teleoperators. An

example of bilateral control is a joystick that provides mechanical resistance to a human operator when a robot makes contact with objects or obstacles in its environment [21].

A number of recent teleoperation systems enable the operator to demonstrate desired control through pose data collected from depth cameras [22]. These types of teleoperation interfaces are unilateral and have been primarily developed to faithfully reproduce the operator's behaviors [12, 23]. Some use gesture recognition as a command signaling mechanism to improve teleoperation [24]. The approach developed in this work differs though in that it infers intentions rather than recognizing pre-programmed directives from depth camera data.

Unfortunately, teleoperation using depth cameras can be difficult for human operators. This is due to noisy sensor or tracking output from depth cameras, poor translation from human input to robotic output, and latency between the human input and robot output [21,25]. Nevertheless, these interfaces are appealing from a cost and flexibility standpoint. A significant research direction has investigated assistive teleoperation [8], in which control is achieved through shared autonomy. One example is teleoperation for free-form tasks using mouse cursor input, in which the task is inferred and used to optimize low-level robotic motions [26]. A recent approach in the brain-computer interface (BCI) teleoperation domain [9] has attempted to address these type of teleoperation issues by infusing a correcting assist action, $\mathbf{A_{assist}}$, to a teleoperation control action, $\mathbf{A_{tel}}$. The exact $\mathbf{A_{assist}}$ value and the degree in which it is added to $\mathbf{A_{tel}}$ is based off of the predicted intention of the teleoperation action. This assisting action addition was shown in [9] to improve task completion metrics in robot teleoperation applications.

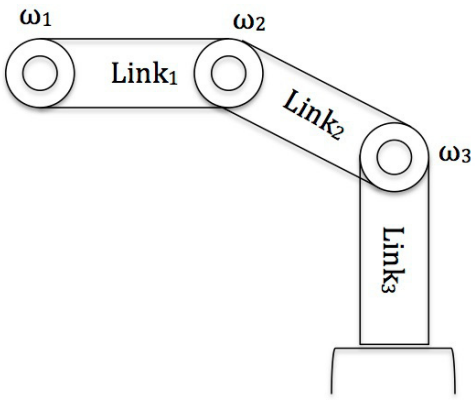

Figure 7. Example of a 3 DOF robot arm. ω_i denotes the joint angle of joint i

2.3 Robotic Arm Systems and Forward/Inverse Kinematics

Robotic arm systems are typically viewed as a system of joints and rigid bodies known as links [27]. As one joint moves in an arm system, the translation and rotation of the links of the system change depending on the geometric configuration of the arm. The Degrees of Freedom (DOF) of an arm system is number of joints in the arm system. Figure 7 shows an example 3 DOF robotic arm system.

The control of motion of a robotic arm system is typically defined by movements of robotic arm joints [27]. Specifically, motion control schemes are used that manipulate joint angles depending on desired joint position, joint velocity, torque applied, etc. [27]. Joint control approaches are natural approaches to arm control as the manipulation elements of a robotic arm system are servomotors in the joints of the arm [27].

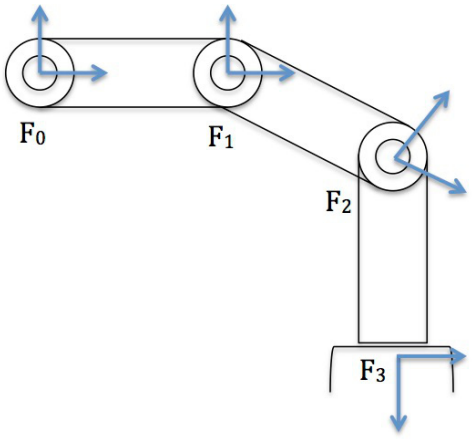

Figure 8. End-effector space of the Figure 7 example. F_0 is the origin coordinate frame and F_3 is the end-effector coordinate frame. F_1 and F_2 are intermediate coordinate frames.

Although motion control is handled in the joint space of a robotic arm, planning out the motion of the arm is difficult using the joint space [27]. This is because one joint's movement can affect any of the translation and rotation of the rigid body links "downstream" of the joint in the arm system. To simplify the complexities of planning out arm motion, the notion of an arm end-effector is used. An arm end-effector is an imaginary point at the end of a robotic arm. The end-effector's translation and rotation are referenced from some origin point, e.g. center of mass of the associated robot. Figure 8 shows the end-effector of Figure 7. Motion planning of the arm can be achieved by first planning out an end-effector's trajectory and then finding a sequence of joint movements to achieve the desired end-effector trajectory.

Forward kinematics refers to converting from the joint angles of a robot arm to the end-effector location and rotation, i.e. from joint space to end-effector space. The forward kinematics problem is often expressed in terms of a 4x4 transform matrix [27]:

$$ {}^{i}T_{j} = \begin{bmatrix} r_{xx} & r_{xy} & r_{xz} & {}^{i}p_{j}^{x} \\ r_{yx} & r_{yy} & r_{yz} & {}^{i}p_{j}^{y} \\ r_{zx} & r_{zy} & r_{zz} & {}^{i}p_{j}^{z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2} $$

where ${}^{i}T_{j}$ denotes the transform matrix between the coordinate frame i and j, the r elements denote the rotation between the coordinate frame i and j between specific translation dimension, and the p elements denote translational offsets from coordinate frame i to j [27]. Both the r and p elements of the transform matrix are functions of the current joint angles. Relating the joint angles to r and p depends on the geometric configuration of an arm system and can be determined using methods in standard robotics sources, for example [27], or supplied by a robot manufacturer.

Relating this transform matrix back to the forward kinematics problem, if frame i is the origin frame of the robotic system and frame j is the end-effector frame, an end-effector's translation and rotation can be extracted from the transform matrix. In Figure 8, the forward kinematics results can be extracted from the ${}^{0}T_{3}$ transform matrix. The p elements of the transformation matrix are the x,y,z translations of the end-effector as the origin frame is the zero point of the arm system. The rotation of the end-effector is typically expressed in a more

compact representation than the rotation elements r directly [27]. There are many options available, but the quaternion rotation representation of w,x,y,z is often used in this work. The w,x,y,z rotation dimensions can be related back to r elements of the transform matrix by [1]:

$$\begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{yx} & r_{yy} & r_{yz} \\ r_{zx} & r_{zy} & r_{zz} \end{bmatrix} = \begin{bmatrix} w^2 + x^2 - y^2 - z^2 & 2(xy - wz) & 2(xz + wy) \\ 2(xy + wz) & w^2 - x^2 + y^2 - z^2 & 2(yz - wx) \\ 2(xz - wy) & 2(yz + wx) & w^2 - x^2 - y^2 + z^2 \end{bmatrix} \tag{2.3}$$

where $||(w, x, y, x)|| = 1$.

Although forward kinematics is useful, it does not address translating an arm's end-effector trajectory into joint movements. To do so requires reversing forward kinematics that is referred to as inverse kinematics. Inverse kinematics is much more difficult to solve than forward kinematics. The difficulty with inverse kinematics is that there are typically many, if not an infinite number of, joint configurations that lead to the exact same end-effector position [27].

To understand this, consider an end-effector that is specified in x,y,z translational dimensions and three rotational angles (it is possible to represent 3-d rotation in only three rotation dimensions rather than the quaternion w,x,y,z dimensions; Euler angles for example [27]). This end-effector is specified with 6 DOF (three translational and three rotational values). The example arm in Figure 7 and Figure 8 is a 3 DOF arm. In this case, the inverse kinematics problem

---

[1] `http://www.cprogramming.com/tutorial/3d/quaternions.html`

is said to be under-specified for the Figure 7 and Figure 8 example because there are less arm variables, i.e. joints, than the specified end-effector. When inverse kinematics is underspecified for an arm system, there are no closed form solutions to inverse kinematics [27]. Now consider a 6 DOF robotic arm, In this case, the inverse kinematics problem is said to be specified and does have many closed form solutions. To understand why there are many solutions, arm joint angles are related to an end-effector through trigonometric functions (sin, cos, tan, etc.). Going from an end-effector to joint angles requires inverting these trigonometric functions (arcsin, arccos, arctan, etc.) Inverse trigonometric functions yield multiple joint angles. This is the root of the multiple closed form solutions in this case. Lastly, consider a robotic arm with more than 6 DOF. In this case, some of the joint angles are redundant for the inverse kinematics calculation which can lead to an infinite number of solutions [27].

Because of the range of possible solutions to an inverse kinematics problem, numerical techniques are frequently used to find approximately optimal solutions considering the current arm joint angles to the desired locations. Numerical inverse kinematics techniques continue to be an area of active research. For example, IKFast, developed recently in [28], is a popular general solution framework for inverse kinematics. Robot manufacturers frequently developed their own techniques using heuristics, etc. for a particular robot.

2.4 Inverse Optimal Control

Inverse optimal control (also known as inverse reinforcement learning) [29–31] seeks a reward or cost function that rationalizes demonstrated control sequences [31]. Though ill-posed in its simplest formulation [30], extensions that seek to provide predictive guarantees create a

well-defined machine learning task [32]. Maximum entropy inverse optimal control [33], for example, seeks to provide robust predictions of the control policy under the logarithmic loss, while learning parameters that define a cost function for inverse reinforcement learning purposes.

In this work, we leveraged extensions of maximum entropy inverse optimal control in linear-quadratic control (LQR) settings [16,34,35]. In these settings, it is assumed that the dynamics of a system being investigated can be represented by a continuous linear state-space representation,

$$\mathbf{s}_{t+1} = \mathbf{A}\mathbf{s}_t + \mathbf{B}\mathbf{a}_t + \epsilon_t, \tag{2.4}$$

where $\mathbf{s_t}$ denotes the state of the system at time t, $\mathbf{a_t}$ denotes the action at time t, ϵ_t denotes some zero mean Gaussian noise, and $\mathbf{A}$ and $\mathbf{B}$ define the system dynamics.

A state-action cost function,

$$\mathsf{cost}(\mathbf{s}_t, \mathbf{a}_t) = \begin{bmatrix} \mathbf{a}_t \\ \\ \mathbf{s}_t \end{bmatrix}^{\mathsf{T}} \mathbf{M} \begin{bmatrix} \mathbf{a}_t \\ \\ \mathbf{s}_t \end{bmatrix}, t < \mathsf{T}, \tag{2.5}$$

and a final state cost penalizing the final state, $\mathbf{s_T}$, from deviating from the desired target, $\mathbf{s_G}$,

$$\mathsf{cost}(\mathbf{s_T}) = (\mathbf{s_T} - \mathbf{s_G})^{\mathsf{T}} \mathbf{M_f} (\mathbf{s_T} - \mathbf{s_G}), \tag{2.6}$$

are learned by updating the $\mathbf{M}$ and $\mathbf{M_f}$ coefficient matrices through demonstrated behaviors using the principle of maximum casual entropy [34]. Specifically, this is done by solving the constrained optimization problem maximizing the casual entropy [16],

$$H(\mathbf{a}\|\mathbf{s}) \triangleq \mathbb{E}_{\hat{\pi}}\left[-\sum_{t=1}^{T}\log\hat{\pi}(\mathbf{a}\|\mathbf{s})\right],$$

such that the predictive policy distribution, $\hat{\pi}(\mathbf{a}\|\mathbf{s}) = \pi(\mathbf{a}_1|\mathbf{s}_1)\pi(\mathbf{a}_2|\mathbf{s}_2)\cdots\pi(\mathbf{a}_T|\mathbf{s}_T)$, matches the demonstrated quadratic state properties, $\tilde{\pi}$, in feature expectation through the following optimization constraints,

$$\mathbb{E}_{\hat{\pi}}\left[\sum_{t=1}^{T-1}\begin{bmatrix}\mathbf{a}_t\\ \mathbf{s}_t\end{bmatrix}\begin{bmatrix}\mathbf{a}_t\\ \mathbf{s}_t\end{bmatrix}^T\right] = \mathbb{E}_{\tilde{\pi}}\left[\sum_{t=1}^{T-1}\begin{bmatrix}\mathbf{a}_t\\ \mathbf{s}_t\end{bmatrix}\begin{bmatrix}\mathbf{a}_t\\ \mathbf{s}_t\end{bmatrix}^T\right], \text{ and}$$

$$\mathbb{E}_{\hat{\pi}}\Big[(\mathbf{s}_T-\mathbf{s}_G)(\mathbf{s}_T-\mathbf{s}_G)^T\Big] = \mathbb{E}_{\tilde{\pi}}\Big[(\mathbf{s}_T-\mathbf{s}_G)(\mathbf{s}_T-\mathbf{s}_G)^T\Big].$$

This optimization allows the state-conditioned probabilistic policy, $\hat{\pi}$, to be formed using the following recursively defined equations,

$$\hat{\pi}(\mathbf{a}_t|\mathbf{s}_t) = e^{Q(\mathbf{s}_t,\mathbf{a}_t)-V(\mathbf{s}_t)}, \tag{2.7}$$

$$Q(\mathbf{s}_t,\mathbf{a}_t) = \mathbb{E}_{\tau(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)}[V(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t)] + cost(\mathbf{s}_t,\mathbf{a}_t), \tag{2.8}$$

$$V(\mathbf{s}_t) = \begin{cases}\underset{\mathbf{a}_t}{\text{softmax}}\, Q(\mathbf{s}_t,\mathbf{a}_t), & t < T\\ (\mathbf{s}_t-\mathbf{s}_G)^T\mathbf{M_f}(\mathbf{s}_t-\mathbf{s}_G), & t = T,\end{cases} \tag{2.9}$$

where the policy distribution is penalized for deviating from the desired goal location, $\mathbf{s}_G$, at time T and the softmax function is a smoothed interpolation of the maximum function, $\underset{\mathbf{x}}{\text{softmax}} f(x) = \log \int_x e^{f(x)} dx$.

After training is complete, (Equation 2.7) through (Equation 2.9) allow the probability of each possible target being the desired goal of an observed partial trajectory to be estimated [16].

CHAPTER 3

LEARNING HUMAN-ROBOT POSE CORRESPONDENCE

3.1 Setup

We used a Microsoft Kinect to obtain point cloud data of a human teleoperator and applied the OpenNI framework to the Kinect data to overlay a digital skeleton model on the human teleoperator's captured depth camera data. The OpenNI Skeleton model has 105 data points, 15 skeleton points each having x,y,z translation and w,x,y,z rotational data. We used these 105 data points (shown in Figure 9) as features, denoted as $\mathbf{X_{OpenNI}}$, to build a correspondence to a Rethink Robotics Baxter Research Robot's[1] two arm joint positions. The Baxter Robot arms are 7 DOF arms, the joint labels are shown in Figure 10. The joint values are denoted as:

$$\mathbf{Y_{joints}} = [s_0, s_1, e_0, e_1, w_0, w_1, w_2]^T \tag{3.1}$$

We assumed the correspondence from $\mathbf{X_{OpenNI}}$ to $\mathbf{Y_{joints}}$ is a linear relationship:

$$\mathbf{Y_{joints}} = \mathbf{X_{OpenNI}^T}\Theta + \epsilon \tag{3.2}$$

where Θ is a coefficient matrix and ϵ is some zero mean Gaussian noise, i.e. $\epsilon \sim N(0, \Sigma^2)$.

Given some OpenNI data, we predicted the corresponding robot joints through regression:

---

[1] http://sdk.rethinkrobotics.com/wiki/Main_Page

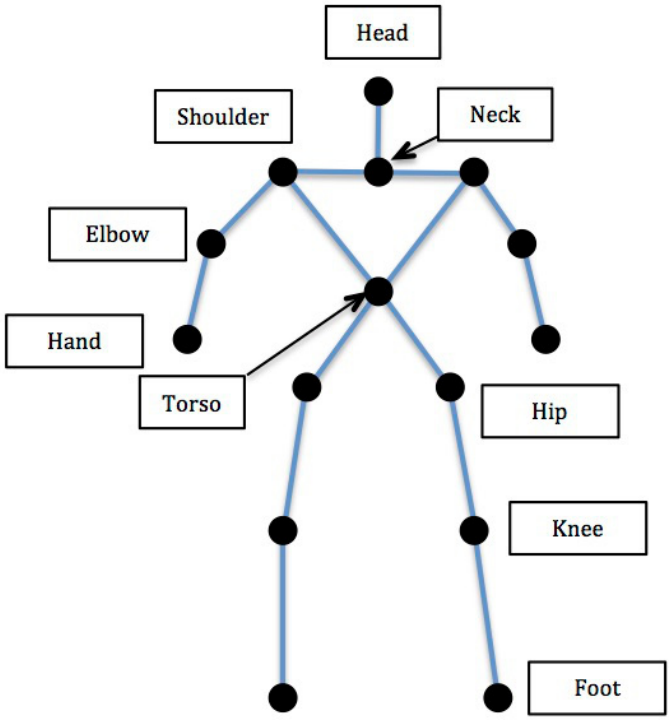

Figure 9. OpenNI skeleton model. Each data point shown has x,y,z translational and w,x,y,z rotational data.

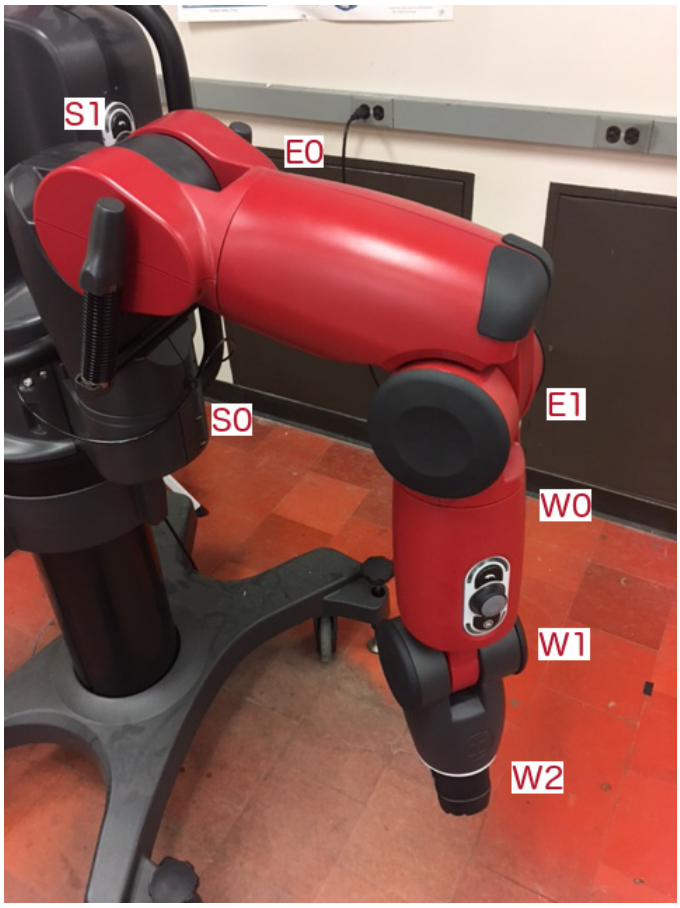

Figure 10. Baxter robot seven joints.

$$\hat{\mathbf{Y}}^{\mathbf{T}}_{\mathbf{joints}} = \mathbf{X}^{\mathbf{T}}_{\mathbf{OpenNI}}\hat{\Theta} \tag{3.3}$$

where $\hat{\mathbf{Y}}^{\mathbf{T}}_{\mathbf{joints}}$ denotes predicted joint values and $\hat{\Theta}$ is the estimated coefficient matrix.

3.2 Training

We learned $\hat{\Theta}$ for each robot arm by collecting data from the Baxter robot as a demonstrator moves the robot arms in "zero-gravity" mode. The arm movements started from a neutral arm position to a final position. While a demonstrator moves one robot arm, another demonstrator mimicked the robot's arm motions with his/her arms and tried to stay synchronized with the robot arm movements. Both the robot arm joints and the corresponding OpenNI Skeleton data were recorded together. We varied the demonstrators to ensure a generalized correlation between human teleoperators and arm joints.

$\hat{\Theta}$ is determined by:

$$\min_{\hat{\Theta}} \sum (\mathbf{Y}^{\mathbf{T}}_{\mathbf{joints}} - \hat{\mathbf{Y}}^{\mathbf{T}}_{\mathbf{joints}})^2 + \lambda \sum |\hat{\Theta}| \tag{3.4}$$

where λ is a regularization coefficient that penalizes the complexity of the models to increase the generality of $\hat{\Theta}$. As λ increases, $\hat{\Theta}$ terms will be driven towards zero [36]. Optimization of Equation 3.4[1] results in $\hat{\Theta}$.

Our final training data set consisted of 11,935 examples for the right arm model and 10,770 examples for the left arm model.

---

[1] we used the sci-kit learn python package to solve for $\hat{\Theta}$: `http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html`

3.3 Results and Discussion

For the final fit, we used 30% of the collected data as a validation set for the correspondence model. We assessed the fitted correspondence model's prediction abilities by the squared error between the predicted joint values and actual joint values squared. The final training error results are shown in Table I and Table II. The right and left arm models performed best when $\lambda = 0$.

Evident by the best performing λ in Table I and Table II, regularization did not the improve models' validation variances. This means regularization during training was a hindrance to the performance of the trained correspondence models. This likely means that enough data was collected to generalize the correspondence models without regularization.

Table III and Table IV show abbreviated parameter training results. The magnitudes of these parameters cannot be used by themselves to determine the importance of any one of the OpenNI values to a robot arm joint. This is because magnitudes of each OpenNI data point are different which affect the magnitudes of the associated parameters.

TABLE I

Right Arm Correspondence Validation Error

Joint	$\lambda = 0.1$	$\lambda = 0.01$	$\lambda = 0.001$	$\lambda = 0$ [1]
s_0	0.126	0.073	0.058	0.043
s_1	0.041	0.021	0.016	0.013
e_0	0.049	0.032	0.019	0.015
e_1	0.157	0.093	0.075	0.068
w_0	0.147	0.108	0.094	0.075
w_1	0.022	0.017	0.014	0.012
w_2	0.050	0.036	0.024	0.017
average	0.084	0.054	0.043	0.035

[1] $\lambda = 0$ corresponds to a standard linear regression
[2] units of reported values are rad^2

TABLE II

Left Arm Correspondence Validation Error

Joint	$\lambda = 0.1$	$\lambda = 0.01$	$\lambda = 0.001$	$\lambda = 0$ [1]
s_0	0.091	0.068	0.040	0.028
s_1	0.049	0.027	0.018	0.013
e_0	0.039	0.020	0.012	0.001
e_1	0.099	0.077	0.057	0.051
w_0	0.067	0.039	0.028	0.023
w_1	0.032	0.020	0.015	0.014
w_2	0.000	0.000	0.000	0.000
average	0.054	0.036	0.024	0.020

[1] $\lambda = 0$ corresponds to a standard linear regression
[2] units of reported values are rad^2

TABLE III

Abbreviated Correspondence Model Parameters for Right Robot S1 Joint

	Torso[1]	Right Shoulder [1]	Right Elbow [1]	Right Hand [1]
$x_{translation}$	1.63	0.05	0.17	0.58
$y_{translation}$	0.42	0.12	0.14	0.31
$z_{translation}$	0.96	0.12	0.06	0.96
$w_{rotation}$	0.22	0.86	0.83	22320
$x_{rotation}$	1.14	0.26	0.86	39923
$y_{rotation}$	0.45	0.38	0.36	22289
$z_{rotation}$	0.93	1.12	0.73	39922

[1] from OpenNI framework skeleton model
[2] only shows 28 out the 105 parameters needed to predict right S1 joint correspondence

TABLE IV

Abbreviated Correspondence Model Parameters for Left Robot S1 Joint

	Torso [1]	Left Shoulder [1]	Left Elbow [1]	Left Hand [1]
$x_{translation}$	1.07	0.20	0.46	0.94
$y_{translation}$	0.43	0.10	0.09	0.39
$z_{translation}$	2.21	0.01	0.19	3.61
$w_{rotation}$	0.30	0.20	0.33	7358
$x_{rotation}$	0.17	1.17	0.01	12824
$y_{rotation}$	0.03	1.11	0.42	7369
$z_{rotation}$	0.09	0.23	0.12	12822

[1] from OpenNI framework skeleton model
[2] only shows 28 out the 105 parameters needed to predict left S1 joint correspondence

CHAPTER 4

GOAL PREDICTION VIA INVERSE LINEAR-QUADRATIC REGULATION

4.1 Setup

We defined a goal as a location in x,y,z translational space that may be reached by the robot arm end-effector. As discussed in Chapter 2, the end-effector is the imaginary endpoint of the robot arm, which can be calculated through the arm's geometry using forward kinematics [37]. The end-effector has both x,y,z translational and w,x,y,z, rotational dimensions referenced from the associated robot's coordinate frame - however, we only considered the translational dimensions of the end-effector for the goal prediction task in this work. End-effector trajectories, as opposed to arm joint angle movements, were used for this goal prediction task as end-effector movements can be modeled easily as a Linear-Quadratic Regulation problem [16].

Our approach to goal prediction was to predict the intent of an end-effector trajectory through space in reference to a goal position. Following the approach outlined in [16], we assumed the linear dynamics of (Equation 2.4), in which the state of the end-effector is defined as:

$$\mathbf{s_t} = [x_t, y_t, z_t, \dot{x}_t, \dot{y}_t, \dot{z}_t, \ddot{x}_t, \ddot{y}_t, \ddot{z}_t, 1]^T, \tag{4.1}$$

and end-effector actions as:

$$\mathbf{a_t} = [\dot{x}_t, \dot{y}_t, \dot{z}_t]^T, \tag{4.2}$$

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 2.5). Additionally, we represented the goal state i of the end-effector using only the goal's translational position,

$$\mathbf{s_{G_i}} = [x_{G_i}, y_{G_i}, z_{G_i}, 0, 0, 0, 0, 0, 0, 0]^T. \tag{4.3}$$

4.2 Cost Matrices Learning

We learned the cost matrix coefficients $\mathbf{M}$ and $\mathbf{M_f}$ from end-effector position data where a human demonstrator moved the robot arms in "zero-gravity" mode from a neutral start position to a goal position. We combined both right and left arm trajectory data into one dataset to train one set of cost matrices. This is because we found that the resulting inverse LQR model generalized to both arms. Our final training consisted of 404 trajectories.

We learned the cost matrices by maximizing causal entropy [34] using gradient descent with an adaptive learning rate [16]. The results are shown in Figure 11 and Figure 12.

There are a lot of zero elements in the learned cost matrices, specifically the velocity and acceleration terms. This was deliberately done; by forcing the solver to keep these elements as zero, the resulting cost matrices generalize across different system orientations and setups [16].

4.3 Goal Likelihood Estimation

From the learned cost matrices, we inferred probabilities of different possible goal states that the operator may want the end-effector to be in given the observed partial trajectory of the

$$
\begin{bmatrix}
-1.61 & 0.96 & -0.98 & 0.00 & 0.00 & 0.00 & -1.02 & 0.97 & -0.98 & -1.33 \\
0.54 & -0.62 & -1.69 \\
0.96 & -2.19 & 0.95 & 0.00 & 0.00 & 0.00 & 0.97 & -1.29 & 0.97 & 0.49 \\
-4.35 & 0.47 & 0.25 \\
-0.98 & 0.95 & -1.63 & 0.00 & 0.00 & 0.00 & -0.98 & 0.97 & -1.03 & -0.62 \\
0.52 & -1.48 & -1.19 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 \\
-1.02 & 0.97 & -0.98 & 0.00 & 0.00 & 0.00 & -1.61 & 0.96 & -0.98 & -2.44 \\
0.51 & -0.57 & -0.16 \\
0.97 & -1.29 & 0.97 & 0.00 & 0.00 & 0.00 & 0.96 & -2.06 & 0.96 & 0.51 \\
-10.6 & 0.50 & 0.13 \\
-0.98 & 0.97 & -1.02 & 0.00 & 0.00 & 0.00 & -0.98 & 0.96 & -1.63 & -0.57 \\
0.50 & -2.88 & -0.16 \\
-1.33 & 0.49 & -0.62 & 0.00 & 0.00 & 0.00 & -2.44 & 0.51 & -0.57 & -5.41 \\
-0.24 & 0.12 & 0.19 \\
0.54 & -4.35 & 0.52 & 0.00 & 0.00 & 0.00 & 0.51 & -10.6 & 0.50 & -0.24 \\
-25.9 & -0.26 & -0.14 \\
-0.62 & 0.47 & -1.48 & 0.00 & 0.00 & 0.00 & -0.57 & 0.50 & -2.88 & 0.12 \\
-0.26 & -6.56 & 0.20
\end{bmatrix}
$$

Figure 11. M Cost Matrix Training Results. Used for both arm predictions. Size is 13 x 13.

$$
\begin{bmatrix}
-1.17 & 0.94 & -0.77 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -1.89 \\
0.94 & -1.39 & 0.97 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.44 \\
-0.77 & 0.97 & -1.26 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -1.39 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
-1.89 & 0.44 & -1.39 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00
\end{bmatrix}
$$

Figure 12. M_f Cost Matrix Training Results. Used for both arm predictions. Size is 10 x 10.

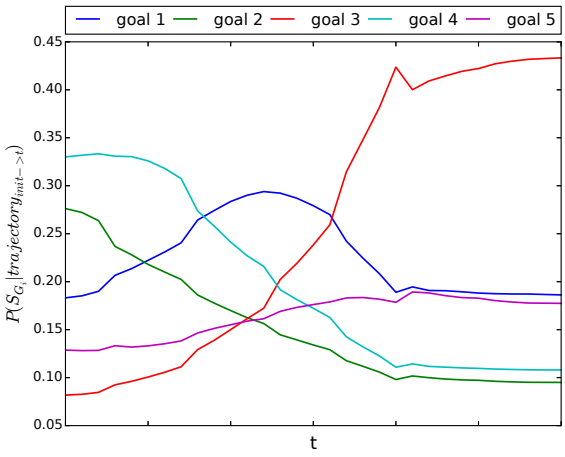

Figure 13. The predicted goal probabilities from the trained inverse LQR model over time for a sequence of robotic arm moves toward a goal.

end-effector in real time. These goal state probabilities are defined as $\mathrm{P}(\mathbf{s_{G_i}}|\mathtt{trajectory}_{\mathrm{init}\to\mathrm{t}})$ and the probability of the most likely intended goal of the partial trajectory, I, as:

$$\mathrm{I} = \max_{\mathrm{i}} \mathrm{P}(\mathbf{s_{G_i}}|\ \mathrm{trajectory}_{\mathrm{init}\to\mathrm{t}}). \tag{4.4}$$

From [16], $\mathrm{P}(\mathbf{s_{G_i}}|\mathtt{trajectory}_{\mathrm{init}\to\mathrm{t}})$ can be calculated through:

$$\mathrm{P}(\mathbf{s_{G_i}}|\mathtt{trajectory}_{\mathrm{init}\to\mathrm{t}}) \propto \prod_{1}^{\mathrm{t}} \pi(\mathbf{a_t}|\mathbf{s_t}, \mathbf{s_{G_i}})\mathrm{P}(\mathbf{s_{G_i}}|\mathbf{s_t}) \tag{4.5}$$

where $\pi(\mathbf{a_t}|\mathbf{s_t}, \mathbf{s_{G_i}})$ is calculated through the learned cost matrices and previous recursive equations, and $\mathrm{P}(\mathbf{s_{G_i}}|\mathbf{s_t})$ can be thought of as the prior probability distribution of goal states

given the partial trajectory alone. In this work, a distance prior was used for $\mathsf{P}(\mathbf{s_{G_i}}|\mathbf{s_t})$, similar to the one used in [16]:

$$\mathsf{P}(\mathbf{s_{G_i}}|\mathbf{s_t}) \propto e^{-\beta \mathtt{dist}(\mathbf{s_t},\mathbf{s_{G_i}})}, \tag{4.6}$$

where $\mathtt{dist}(\mathbf{s_t},\mathbf{s_{G_i}})$ is a function that computes the Euclidean distance between the spatial coordinates of $\mathbf{s_t}$ and $\mathbf{s_{G_i}}$, and β is an adjustable coefficient that increases the importance of distance on the distribution. As $\mathtt{dist}(\mathbf{s_t},\mathbf{s_{G_i}})$ decreases, $\mathsf{P}(\mathbf{s_{G_i}}|\mathbf{s_t})$ increases effectively making closer targets more probable. Figure 13 depicts the predicted goal probability from the trained model change over time as an arm moves toward a specific goal.

CHAPTER 5

GOAL-BASED CONTROL ASSISTANCE

5.1 Setup

Using the goal predictions model from Chapter 4, we adjusted the level of autonomy between human and autonomous control. The specific approach, employed from [9, 38], was to consider an action taken by a robot at time t, $\mathbf{A_{r,t}}$, to be a linear combination of a teleoperation action, $\mathbf{A_{tel,t}}$ and an assisting action, $\mathbf{A_{assist,t}}$:

$$\mathbf{A_{r,t}} = \alpha \mathbf{A_{tel,t}} + (1 - \alpha)\mathbf{A_{assist,t}} \tag{5.1}$$

where α is a mixing coefficient. When α is close to 1, $\mathbf{A_{r,t}}$ is mainly a function of the teleoperation translated action $\mathbf{A_{tel,t}}$. As alpha decreases towards 0, $\mathbf{A_{r,t}}$ becomes increasingly a function of an assist action $\mathbf{A_{assist,t}}$. By defining α as a function of predicted intent [16], a noisy teleoperation action can be corrected by mixing in an appropriate assisting action that reflects the true intention of the teleoperation action.

For this work, we used an arm joint angle based position controller[1]. Because of this, the denoted action variables in (Equation 5.1) for this work were arm joint positions. Specifically, $\mathbf{A_{tel,t}}$ was the joint values from the linear correspondence model trained from a human tele-

[1] `http://sdk.rethinkrobotics.com/wiki/Arm_Control_Modes`

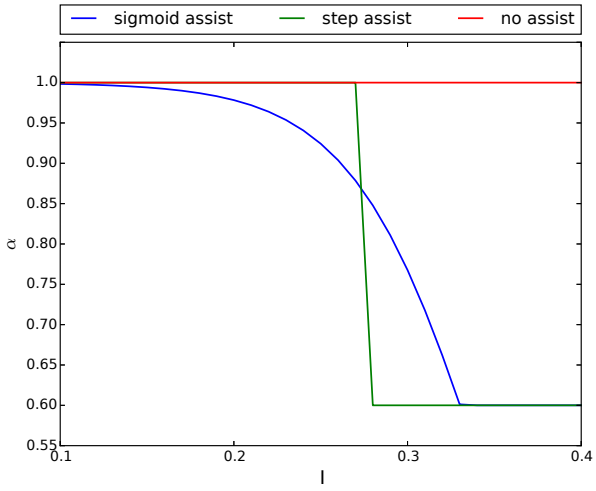

Figure 14. Three α selection methods after final parameter adjustments.

operator to robot arm joints discussed in Chapter 3. $\mathbf{A_{assist,t}}$ was calculated by finding the end-effector goal with the highest probability from the goal prediction model and applying inverse kinematics[1] to the goal position with the current arm joint positions as a seed to the inverse kinematic calculations. Then, $\mathbf{A_{tel,t}}$ and $\mathbf{A_{assist,t}}$ were linearly mixed together per (Equation 5.1) to form $\mathbf{A_{r,t}}$ which was used by the joint position arm controller to manipulate the robot arms.

5.2 α Mixing Strategies

We considered three different approaches for adjusting the value of the mixing coefficient α in /eqrefaction$_{\text{m}}$ixing.Thefirstapproachwasanoassistapproachwhereα is 1.0 for all I values.

[1] http://sdk.rethinkrobotics.com/wiki/IK_Service_Example

As α remains at 1.0 regardless of I, this method prevented any assisting action from being applied to a teleoperation control action. For this reason, we refer to this α approach in this work as the *no assist* approach in this work.

The second approach varied α as a sigmoid function of I [9]:

$$\alpha = \frac{1}{1 + e^{-a(1-I)+o}} \tag{5.2}$$

where a and o are adjustable parameters. Because this approach varied α as I varied, the approach resulted in an $\mathbf{A_{r,t}}$ that is a linear mixture of an assisting action and a teleoperation control action. We refer to this α approach in this work as the sigmoid assist approach.

The last approach we considered was α as a step function of I [8]:

$$\alpha = \begin{cases} 1.0, \text{ if } I < I_{threshold} \\ c, \text{ else} \end{cases} \tag{5.3}$$

where $I_{threshold}$ and c are adjustable parameters. Like the sigmoid assist, this method did mix in an assisting action with a teleoperation control action. We refer to this α approach in this work as the step assist approach. Figure 14 shows the three alpha methods as a function of I after our final parameter adjustments.

As discussed in [9], it is important to set an α_{min} where the value of α cannot drop below. If α decreases too much, $A_{assist,t}$ becomes dominant in (Equation 5.1) where if the guessed intent of a partial trajectory is wrong the human teleoperator will never be able to correct the arm

because $A_{tel,t}$ will not be a large enough component in $A_{r,t}$. We set an α_{min} value for this work of 0.60.

CHAPTER 6

VALIDATION EXPERIMENTS

6.1 Setup

To test our approach, we selected ten different end-effector goal locations, five for each of the Baxter arms. The selected goals were different than any of the goals used to train the goal prediction and linear correspondence models. The goals were visually shown in the testing space around the robot with approximately four-inch diameter spheres numbered to reflect the goal index they were associated with. There was a Kinect camera directly in front of the robot testing space setup where the human demonstrator's OpenNI skeleton data was captured. The Kinect camera was positioned so that a demonstrator could see both the Baxter robot and the goals.

For each demonstrator, all three of the α selection criteria were tested twice. The order in which the assist and no assist α selection methods were used was randomized between demonstrators. A demonstrator was never told what α selection method was being used at a given time.

For a given α selection test, all ten goals were tested in a random order. One goal test is referred to as a goal test sequence. A sequence always started with the robot arms at neutral positions that were the same for every sequence. The demonstrator was told what the objective

a. Starting Neutral Position

b. Teleoperate Arm Towards Goal

c. At Goal

d. Results Displayed

Figure 15. The steps of a task in the testing sequence.

goal was prior to the start of any goal test sequence through a graphics display and given five seconds to prepare. After this countdown, the sequence started.

At the start of a test sequence, the associated arm position controller of the selected goal was enabled. The input used by the arm controller was the output from (Equation 5.1), where the α mixing coefficient was based off the associated α mixing strategy. The objective of the sequence was to get the associated arm's end-effector within 0.13 meters of the selected goal, calculated by Euclidean distance. The demonstrator had to keep the end-effector within the distance tolerance for 2.0 seconds to register the sequence as a success. The demonstrator had 15 seconds to complete a sequence, otherwise, a timeout was recorded for the sequence. When using an assist method, the α value was kept at 1.0 until the arm end-effector traveled at least 0.6 meters. This was to allow a partial trajectory to be established for the inverse LQR goal prediction model.

6.2 Results and Discussion

We used the metrics of the completion time and the end-effector distance traveled to evaluate each sequence. The end-effector distance traveled was calculated by summing up the Euclidean distances between controller steps. If a timeout occurs for a sequence, the completion time was recorded as the timeout setting amount, which was 15 seconds.

Overall, we gathered 18 completed testing sets of data from human demonstrators. Each demonstrator contributed 60 data points, 6 α selection tests each having 10 goals.

Averaging the sequence results of all 18 demonstrators (Table V), we observed an average decrease in completion time of 1 to 2 seconds and distance traveled by 0.5 meters or less when

TABLE V

Average Improvement of Completion Time and Distance Traveled

	Δt [s][1]	**p-value**[2]	**Δd [m]**[1]	**p-value**[2]
No Assist to Sigmoid	2.1	2.8e-6	0.47	2.2e-5
No Assist to Step	1.3	5.8e-5	0.29	4.9e-5
Step to Sigmoid	0.8	1.3e-2	0.18	5.7e-5

[1] Average improvement across all participants with both left and right arm results. Results shown are decrease improvements.

[2] The p-values computed using pairwise t-testing for each participant comparing participant's average results from each of the control strategies. Shown p-values are one-sided.

comparing the assist methods to the no assist method. These results might appear to be moderate improvements. However, considering the relatively easy task used to test with, these results are of practical importance.

The sigmoid assist method provided a larger improvement than the step assist method. This result is not surprising given that the sigmoid assist provides a continuous range of α values compared to the two α value outputs of the step assist method. The p-values of these results, calculated using a paired statistical test for different methods using paired teleoperation sequences to each goal, show that these observed improvements are statistically significant.

Averaging the sequence results on a per-goal basis (Figure 16, Figure 17) improvements are seen in both task completion time and distance traveled across all 10 goals when comparing

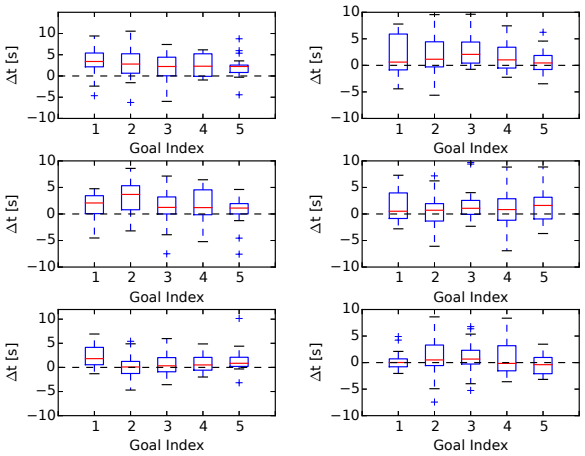

Figure 16. Time improvements results for both arms. Plots in the first row show the improvement from using no assist to using the sigmoid assist, plots in the second row show the improvement from using no assist to using the step assist, and plots in the last show the improvement from using step assist to sigmoid assist. The left column is the left arm results and the right column is the right arm results.

results of the assisting methods and no assist method as well. For a majority of the goals, the sigmoid assist method out performs the step assist method.

We observed the improvements the assist methods provide in the actual arm end-effector trajectories of the test sequences as well. For example, in Figure 18, one demonstrator was able to teleoperate the arm in a much more efficient path to the goal with the assist methods. When no assist was applied, the end-effector trajectory appears more sporadic, taking longer to stabilize near the goal coordinates. Although this is one specific example of improvement, it provides insight into how the assisting actions are actually improving the efficiency of teleoperation.

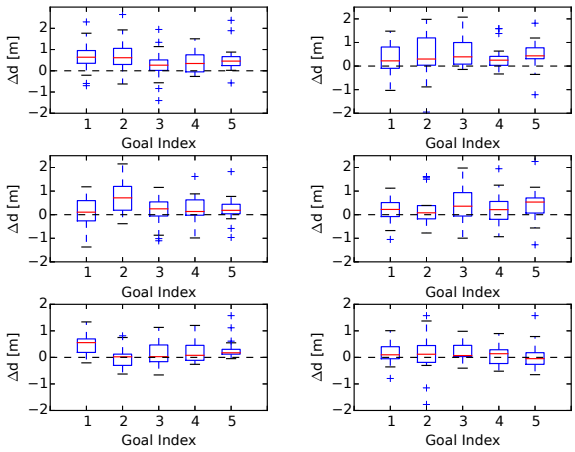

Figure 17. Distance traveled results for both arms. Plots in the first row show the improvement from using no assist to using the sigmoid assist. Plots in the second row show the improvement from using no assist to using the step assist. Plots in the last show the improvement from using step assist to sigmoid assist. The left column contains the left arm results and the right column contains the right arm results.

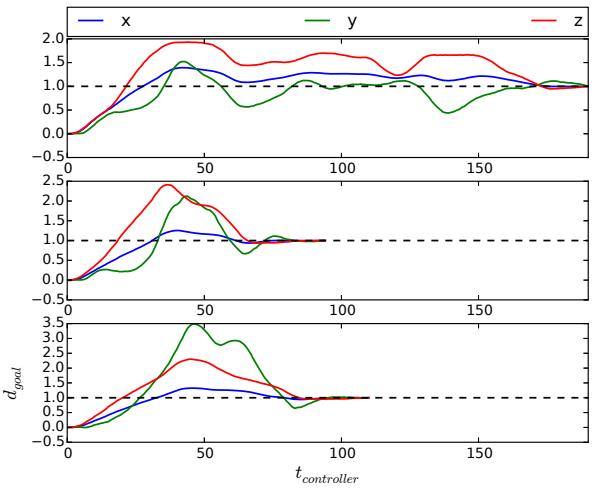

Figure 18. Example of the improvement that the assisting methods provide to the arm end-effector trajectory. Data is from the same participant attempting to teleoperate a robot arm from the neutral position to the same goal position. From the top, the first plot is the no assist method end-effector trajectory result, the second plot is the sigmoid assist method, and the last plot is the step assist method. All three translational dimensions (x,y,z) of an end-effector are shown. The $\mathtt{t}_{\mathtt{controller}}$ axis represents time referenced by the arm controller time step since the start of a testing sequence. The $\mathtt{d}_{\mathtt{goal}}$ axis represents the fraction of the distance the arm end-effector has traveled towards the goal.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

In this work, we showed that inverse optimal control approaches, specifically inverse LQR, can be used to extract the intent of teleoperation control actions in real time. This intent can be used to improve teleoperation task completion efficiency by applying an assist action to a teleoperation control action. Specifically, the results were shown in a depth camera teleoperation setting.

There are many interesting extensions to this work to consider in the future. First, we believe the improvements shown in this work will increase as the difficulty of the teleoperation task increases. Therefore, extending this work to more complicated teleoperation tasks, such as grasping objects and more difficult arm navigation tasks, would show if the improvements this developed teleoperation system provide increase as a function of task complexity.

Second, it is possible to predict the best action to get a robot arm end-effector into a goal state from the inverse LQR model. Instead of using goal coordinates and inverse kinematics for $A_{assist,t}$, we want to use these predicted actions as $A_{assist,t}$ instead.

Lastly, incorporating obstacle avoidance with the assist action would be very beneficial. In [39], it was shown that the inverse LQR prediction method can incorporate waypoint states $s_{W,i}$ into this inverse optimal control formulation with an additional cost term in the cost function shown above. Using this waypoint formulation addition, we believe it is possible, through arm demonstrations that avoid obstacles during training, that a cost function can be

learned that provides next actions that avoid obstacles. With the completion of the second extension just mentioned above, this next action could be used as $A_{\text{assist,t}}$.

APPENDICES

Appendix A

IEEE MANUSCRIPT COPYRIGHT NOTICE

Much of this work has been submitted to IEEE for consideration for ICRA 2017 under the manuscript [40]. My fellow co-authors for this manuscript are Sanket Gaurav [1], Mathew Monfort[2], Lingfei Zhang[1], and Brian D. Ziebart[1].

This thesis document uses excerpts from [40] without citation as this thesis document is an expansion of the manuscript.

Please note: if this submitted manuscript is accepted, IEEE will retain the copyright on all figures, tables, and text excerpts in the accepted manuscript. IEEE's thesis reuse policy is available on the web [3].

Specific figures used in this thesis that are included in the first submitted manuscript are:

- Figure 13
- Figure 14
- Figure 15

---

[1]Sanket Gaurav, Lingfei Zhang, and Brian D. Ziebart are with the Department of Computer Science, University of Illinois at Chicago, 851 S. Morgan St. (M/C 152) Chicago, IL 60607 sgaura2@uic.edu, lzhang44@uic.edu, and bziebart@uic.edu

[2]Mathew Monfort is with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 32 Vassar St, Cambridge, MA 02139 mmonfort@mit.edu

[3]https://www.ieee.org/publications_standards/publications/rights/permissions_faq.pdf

- Table V
- Figure 16
- Figure 17
- Figure 18

CHAPTER 2

TELEOPERATION DEMONSTRATION VIDEO

A demonstration video of the final goal-predictive teleoperation system can be found at `https://youtu.be/czQ7v7OGKkk`.

CITED LITERATURE

1. Nagatani, K., Kiribayashi, S., Okada, Y., Otake, K., Yoshida, K., Tadokoro, S., Nishimura, T., Yoshida, T., Koyanagi, E., Fukushima, M., et al.: Emergency response to the nuclear accident at the fukushima daiichi nuclear power plants using mobile rescue robots. Journal of Field Robotics, 30(1):44–63, 2013.

2. Harada, K., Kajita, S., Saito, H., Morisawa, M., Kanehiro, F., Fujiwara, K., Kaneko, K., and Hirukawa, H.: A humanoid robot carrying a heavy object. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pages 1712–1717. IEEE, 2005.

3. Lendvay, T. S., Hannaford, B., and Satava, R. M.: Future of robotic surgery. The Cancer Journal, 19(2):109–119, 2013.

4. Levine, S., Finn, C., Darrell, T., and Abbeel, P.: End-to-end training of deep visuomotor policies. Journal of Machine Learning Research, 17(39):1–40, 2016.

5. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature, 518(7540):529–533, 2015.

6. Latombe, J.-C.: Robot motion planning, volume 124. Springer Science & Business Media, 2012.

7. 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.

8. Dragan, A. D. and Srinivasa, S. S.: Formalizing assistive teleoperation. MIT Press, July, 2012.

9. 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.: Autonomy infused teleoperation with application to bci manipulation. arXiv preprint arXiv:1503.05451, 2015.

10. Song, W., Guo, X., Jiang, F., Yang, S., Jiang, G., and Shi, Y.: Teleoperation humanoid robot control system based on kinect sensor. In Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2012 4th International Conference on, volume 2, pages 264–267. IEEE, 2012.

11. Du, G., Zhang, P., Mai, J., and Li, Z.: Markerless kinect-based hand tracking for robot teleoperation. International Journal of Advanced Robotic Systems, 9, 2012.

12. Du, G. and Zhang, P.: Markerless human–robot interface for dual robot manipulators using kinect sensor. Robotics and Computer-Integrated Manufacturing, 30(2):150–159, 2014.

13. Moeslund, T. B., Hilton, A., and Krüger, V.: A survey of advances in vision-based human motion capture and analysis. Computer vision and image understanding, 104(2):90–126, 2006.

14. Shingade, A. and Ghotkar, A.: Animation of 3d human model using markerless motion capture applied to sports. arXiv preprint arXiv:1402.2363, 2014.

15. Shiratori, T., Park, H. S., Sigal, L., Sheikh, Y., and Hodgins, J. K.: Motion capture from body-mounted cameras. In ACM Transactions on Graphics (TOG), volume 30, page 31. ACM, 2011.

16. Monfort, M., Liu, A., and Ziebart, B. D.: Intent prediction and trajectory forecasting via predictive inverse linear-quadratic regulation. In AAAI, pages 3672–3678, 2015.

17. Ju, Z., Yang, C., and Ma, H.: Kinematics modeling and experimental verification of baxter robot. In Control Conference (CCC), 2014 33rd Chinese, pages 8518–8523. IEEE, 2014.

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

19. Sarbolandi, H., Lefloch, D., and Kolb, A.: Kinect range sensing: Structured-light versus time-of-flight kinect. Computer Vision and Image Understanding, 139:1–20, 2015.

20. Gulshan, V., Lempitsky, V., and Zisserman, A.: Humanising grabcut: Learning to segment humans using the kinect. In Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on, pages 1127–1133. IEEE, 2011.

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

22. Shotton, J., Girshick, R., Fitzgibbon, A., Sharp, T., Cook, M., Finocchio, M., Moore, R., Kohli, P., Criminisi, A., Kipman, A., et al.: Efficient human pose estimation from single depth images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(12):2821–2840, 2013.

23. Reddivari, H., Yang, C., Ju, Z., Liang, P., Li, Z., and Xu, B.: Teleoperation control of baxter robot using body motion tracking. In Multisensor Fusion and Information Integration for Intelligent Systems (MFI), 2014 International Conference on, pages 1–6. IEEE, 2014.

24. Hu, C., Meng, M. Q., Liu, P. X., and Wang, X.: Visual gesture recognition for human-machine interface of robot teleoperation. In Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, volume 2, pages 1560–1565. IEEE, 2003.

25. Chen, J. Y., Haas, E. C., and Barnes, M. J.: Human performance issues and user interface design for teleoperated robots. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 37(6):1231–1245, 2007.

26. Hauser, K.: Recognition, prediction, and planning for assisted teleoperation of freeform tasks. Autonomous Robots, 35(4):241–254, 2013.

27. Siciliano, B. and Khatib, O.: Springer handbook of robotics. Springer Science & Business Media, 2008.

28. Diankov, R.: Automated construction of robotic manipulation programs. 2010.

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

30. Ng, A. Y., Russell, S. J., et al.: Algorithms for inverse reinforcement learning. In Icml, pages 663–670, 2000.

31. Abbeel, P. and Ng, A. Y.: Apprenticeship learning via inverse reinforcement learning. In Proc. International Conference on Machine Learning, pages 1–8, 2004.

32. Ratliff, N., Bagnell, J. A., and Zinkevich, M.: Maximum margin planning. In Proc. International Conference on Machine Learning, pages 729–736, 2006.

33. 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.

34. 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.

35. Levine, S. and Koltun, V.: Continuous inverse optimal control with locally optimal examples. In International Conference on Machine Learning, 2012.

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

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

38. Dragan, A. D. and Srinivasa, S. S.: A policy-blending formalism for shared control. The International Journal of Robotics Research, 32(7):790–805, 2013.

39. 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. http:// www. ias. informatik. tu-darmstadt. de/ uploads/ ALR2014/ Byravan_ ALR2014. pdf. Citeseer, 2014.

40. Schultz, C., Gaurav, S., Monfort, M., Zhang, L., and Ziebart, B. D.: Goal-predictive robotic teleoperation from noisy sensors. In Submitted to IEEE ICRA 2017 IEEE. IEEE.

VITA

NAME: Christopher Schultz

EDUCATION: B.S., Chemical Engineering, University of Wisconsin - Madison, Madison, Wisconsin, 2011

M.S., Computer Science, University of Illinois at Chicago, Chicago, Illinois, 2016

TEACHING: Department of Computer Science, University of Illinois at Chicago, Chicago, Illinois: Computer Systems Teaching Assistant, 2016

EXPERIENCE: Diadoki, Inc. Chicago, IL Developer, 2015 - 2016

Exelon Generation, Warrenville, IL Developer Intern, 2015 - 2016

Veolia Water Solutions and Technologies, Plainfield, IL Project Engineer, 2012 - 2014

International Titanium Powder, a Cristal Global Company, Lockport, IL

Process Development Engineer, 2011 - 2012

PUBLICATIONS: Schultz, C., Gaurav, S., Monfort, M., Zhang, L., and Ziebart, B. D.: Goal-predictive robotic teleoperation from noisy sensors. Submitted to IEEE ICRA 2017.