

Copyright
by
Matthew William Horn
2017

The Thesis Committee for Matthew William Horn
Certifies that this is the approved version of the following thesis:

**Quantifying Grasp Quality Using an Inverse
Reinforcement Learning Algorithm**

APPROVED BY

SUPERVISING COMMITTEE:

Sheldon Landsberger, Supervisor

Mitch Pryor, Co-Supervisor

**Quantifying Grasp Quality Using an Inverse
Reinforcement Learning Algorithm**

by

Matthew William Horn, B.S.M.E.

THESIS

Presented to the Faculty of the Graduate School of
The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN ENGINEERING

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2017

Dedication

Dedicated to my family. Thank you for supporting this journey.

Acknowledgments

I wish to thank the members of the Nuclear and Applied Robotics Group at UT Austin for their support and good times. I also want to thank my family for motivating and supporting me through my time here at UT Austin.

This material and research is supported by an Integrated University Program Graduate Fellowship provided by the Department of Energy. The fellowship name is part of the Nuclear Energy University Program (NEUP).

Abstract

Quantifying Grasp Quality Using an Inverse Reinforcement Learning Algorithm

Matthew William Horn, M.S.E.

The University of Texas at Austin, 2017

Supervisors: Sheldon Landsberger

Co-Supervisor: Mitch Pryor

This thesis considers the problem of using a learning algorithm to recognize when a mechanical gripper and sensor combination has achieved a robust grasp. Robotic hands are continuously evolving with finer motor control and higher degrees of freedom which can complicate the ability of an operator to determine if a gripper has achieved a successful grasp. Robots working in hazardous environments especially need confirmation of a successful grasp as the cost of failure is often higher than in traditional factory environments. The object set found in a nuclear environment is the focus of this effort. Objects in this environment are typically expensive (or one-of-a-kind), rigid, radioactive (or toxic), dense, and susceptible to dents, scratches, and oxidation. To validate the robustness of a grasp option, an online inverse reinforcement learning approach is evaluated as a method to quantify grasp quality. This approach is

applied to an industrial-grade under-actuated robotic hand equipped with 36 pressure sensors. An expert trains the inverse reinforcement learning algorithm to generate a reward function which scores each grasp so - when combined with fuzzy logic - provides a general success or fail along with a confidence level. Utilizing the trained inverse reinforcement learning algorithm in a glovebox environment reduces the number of potential failing and untrustworthy grasps by scoring executed grasps and rejecting grasps that are similar to prior failed grasps while allowing further execution of movement when a grasp has been scored highly. The trained algorithm incorrectly classified grasps of insufficient quality less than 5% of the time in experimental hardware tests, showing that the algorithm can be applied to the glovebox environment to improve grasp safety. Thus the combination of grasp selection and pressure sensor validation provides a more efficient, robust, and redundant method to assure items can be safely handled during remote automation processes.

Table of Contents

Acknowledgments	v
Abstract	vi
List of Tables	x
List of Figures	xi
Chapter 1. Introduction	1
1.1 Environmental Challenges	3
1.2 Robotic Hands	5
1.3 Grasp Validation	12
1.4 Summary of Objectives	13
1.5 Organization	14
Chapter 2. Literature Review	15
2.1 Grasp Planners	17
2.1.1 Early Grasping Work	17
2.1.2 Other Modern Grasp Efforts	22
2.2 Non-Learning Based Grasp Validation	26
2.3 Introduction to Probabilistic Methods in Robotics	27
2.3.1 Supervised Learning	29
2.3.2 Reinforcement Learning	29
2.3.3 Fuzzy Logic	30
2.3.4 Machine Learning Components	30
2.4 Efforts Involving Learning and Grasp Validation	32
2.5 Summary	33

Chapter 3. Learning Solution and Data Gathering	35
3.1 Iterative Improvement	35
3.2 Inverse Reinforcement Learning Algorithm Design	37
3.2.1 Features	37
3.2.2 Scoring Function	41
3.2.3 Interaction Model	42
3.2.4 Software Application	43
3.3 Data Gathering	44
3.4 Summary	47
Chapter 4. Demonstration and Analysis	48
4.1 Hardware	48
4.1.1 Touch Sensors	49
4.1.2 Robotiq Hardware Sensing Capabilities	52
4.2 Software Framework	55
4.2.1 Robot Operating System (ROS)	56
4.3 Test Objects and Results	58
4.4 Summary of Demonstrations and Analysis	65
Chapter 5. Conclusions and Future Work	67
5.1 Summary of Covered Topics	68
5.2 Application	70
5.3 Future Work	70
5.4 Concluding Comments	73
Bibliography	75
Index	84
Vita	85

List of Tables

3.1	Features and their respective reporting range.	38
4.1	Robotiq table covering the various advantages and disadvantages lab members experienced by employing the Robotiq 3-Finger Adaptive gripper for various applications.	50
4.2	Summary of Robotiq object detection correlation.	54
4.3	Summary of force setting correlation.	55
4.4	Each test object has a total of 55 grasp data sets. 5 data sets per object were reserved for testing IRLA after initial training.	58
4.5	Listed object test set by shape in order of Figure 4.5	60
4.6	Combined results from all saved data, tested with 25 random samples (5 from each shape type). The shape parameter and weighting is ignored when inputting into IRLA.	63
4.7	Summarized training and testing results from objects in Figure 4.5	63

List of Figures

1.1	A typical glovebox that handles hazardous materials [50]. . . .	4
1.2	A typical hot box that employs teleoperation to handle extremely hazardous materials not suitable for a glovebox [42]. .	6
1.3	A vacuum system utilized in moving small cups. The gray portion at the bottom is the actual suction cup.	8
1.4	Images of potential robotic grippers.	9
1.5	Takktile Robotiq components.	12
2.1	Cutkoskys taxonomy of prehension [10]	21
2.2	A primitive shape decomposition of a coffee mug [4]	23
2.3	An SVM sorted optimal grasp generated through GraspIt! [35]	25
2.4	Five objects with three different grasps given by participants on object models [24].	33
3.1	Trajectory Preference Perceptron. w_O and w_E stand for object and environmental features.	36
3.2	A Takktile sensor reading obtained from grasping a rope. The colors represent the range of forces normalized by the maximum force. Green represents 0 - 10% of the maximum force, Yellow 10% - 40%, Orange 40% - 70%, and Red 70% - 100%. (Green = 0, Yellow = 1, Orange = 2, Red = 3)	40
3.3	Representation of active nodes within ROS and their basic communication structure.	44
3.4	Robotiq grasping data collection visualization	45
3.5	Robotiq basic movements and commanded positions	46
4.1	Final Robotiq 3-Finger Adaptive gripper and Takktile sensor suite.	49
4.2	Function scheme presented in Laschi’s paper. [28]	51
4.3	From the paper <i>Inexpensive and Easily Customized Tactile Array Sensors using MEMS Barometers Chips</i> [49]. Sensor output values versus applied surface load for differing rubber thickness.	52

4.4	Sample output from Scoring Control	59
4.5	Object set used for testing and training purposes, arranged by shape. Shapes were chosen based on relationships between the shapes and glovebox tasks and to vary the sizes and properties.	60
4.6	Amorphous test results with visualized fuzzy logic separation of correctly labeled grasps.	61

Chapter 1

Introduction

General grasp validation - the ability to determine the quality of a grasp and its likelihood to fail - is usually a hard-coded application dependent on an unchanging environment and is object-dependent. Most grasping algorithms instead focus on generating grasps using vision-based systems to identify a set of valid grasp points to minimize grasping failure. Once a grasp has been determined to be the best based on those methods, other validation algorithms then verify the grasp using a simulated physical model or by performing a vision-based check on the final grasp configuration. Humans are adept at deciding whether an object will slip out of their hands or if their grasp is precarious on contact. By adding sensors for touch (which encompasses sensations like pressure, temperature, shape, elasticity, sharpness, roughness, etc.), providing an automatic system to self-determine grasp quality will necessarily be complex if programmed by hand and some efforts to do so are reviewed in Chapter 2. One option to avoid such complexity is to apply self-learning techniques which are reviewed in Chapter 3.

A generic grasp quality system is limited due to two concerns: the vast array of differing hardware, and the high costs of said hardware. Other

systems, such as vision systems, act and respond the same way across the spectrum of hardware. Vision systems also have a wide base of support, with many universities and research organizations releasing vision data sets to help further research. Vision system hardware, unlike grasping hardware, is relatively cheap: common 3-D vision systems' costs ranges from \$100 [31] to \$169 [33] but even industrially hardened or military specced hardware is typically less than \$5,000. These two advantages allow vision system algorithms far more use than other hardware based systems due to their availability and low entry cost. However, vision systems can only let an algorithm guess, or model an object and can be difficult to judge grasp quality when an object is contained within a robotic grasp, leading to the need for in-gripper hardware sensing capabilities.

Like vision capabilities, much of grasp generation (i.e. loading physics models into software, determining feasible grasping points, simulating a grasped object's unconstrained motion, etc.) can be simulated. Research on these topics in the community has seen a lot of progress in the past few years; however, due in part to the relative complexity of creating gripper models, friction models, and object models, usage has been restricted to function specific applications. A notable grasp simulator, *GraspIt!* [18], can be employed, but requires a large amount of knowledge and data to be useful to the user, such as the previously mentioned need for specific data points for each object and gripper. The requirement to constantly update objects that need to be grasped limits the applications where these simulators can be used, especially when novel

objects are encountered. A final hurdle for simulated learning for grasping involves the fact that grasping research for one gripper configuration may not transfer to other gripper designs.

1.1 Environmental Challenges

Successfully determining whether a grasp has been accomplished to a satisfactory degree is a challenging problem. The difficulty stems from the diverse array of potential objects to grasp. This large array of objects usually makes a learning approach intractable. The variance in the problem can be alleviated due to environmental factors that affect the focus of this research. The impetus to perform this research is in part to increase worker safety at Los Alamos National Lab (LANL) and other nuclear facilities with specific applications for the handling of hazardous materials inside gloveboxes. The environment poses significant dangers on its own, but also restricts the scope of the objects which makes the problem tractable.

Gloveboxes used by the nuclear industry are designed to house radioactive materials and reduce the radiation exposure to humans as shown in Figure 1.1. In addition to the hazards associated with radioactive materials, other materials may also be present. These materials can be reactive or toxic to human life. In these cases, gloveboxes are sometimes filled with inert Noble gases, such as Argon. These differences in air concentration would normally cause an effect in the barometric sensors embedded in the Takktille sensors, though the unique rubber coating prevents this problem from occurring.

Radiation workers are regulated by the federal government for their health and safety. The federal regulation pertaining to keeping doses *as low as (is) reasonably achievable* is contained in ALARA, Code of Federal Regulations, title 10, sec 20.1003. Developing applications for this domain constrains the focus of this application to training and testing outside of a radioactive area.



Figure 1.1: A typical glovebox that handles hazardous materials [50].

In addition to concerns over ALARA, gloveboxes pose other challenges in regards to the space itself. Gloveboxes differ from hot boxes, shown in

Figure 1.2. Gloveboxes are designed primarily for human operators while hot boxes may contain teleoperated robots to handle especially dangerous materials. The application this paper develops can be used at either location, though the primary focus will be within a glovebox for automating tasks. To facilitate human handling and manipulation of materials, gloveboxes are small with many tools and objects located within, though safety restricts efficient usage of the available workspace. This constrained environment requires precise movements, and a grasp that fails halfway through a motion may have explosive or harmful results.

Even though it poses considerable challenges, the glovebox environment restricts the object set which has the potential to make the learning problem tractable. The set of items allowed inside of a glovebox are well-known and thoroughly studied. The objects inside of the glovebox are selected or designed for their ergonomic form and operator lifting restrictions. The algorithms developed here are also applicable for industrial applications due to the limited number of objects that a robot will manipulate in that environment.

1.2 Robotic Hands

Grasping is a complex task involving both hardware and software, and depending on the implementation, can be extremely time consuming to implement. To reduce the need for verifying object properties, models, and visual recognition setup, this effort will focus on improving grasping software. The sections below will briefly review the state-of-the-art in grasping hardware and



Figure 1.2: A typical hot box that employs teleoperation to handle extremely hazardous materials not suitable for a glovebox [42].

discuss which gripper(s) will be used as a part of this effort. The proposed efforts will be hardware agnostic to accommodate the varied nature of grasping technologies, and so the actual gripper choice does not have a large impact on the final efforts detailed in this thesis.

Many robotic hands have little resemblance to human hands, usually missing fingers, joints, and sensing capabilities. Many do not resemble "hands"

at all, leading to difficulties for applying a human knowledge of grasping to robotic grippers. Robotic hands are often designed or adapted for specific tasks, and in those tasks they outperform human hands in speed and precision. These systems are usually more robust than human hands but lack the flexibility and range of movement. Because the algorithm explored is intrinsically hardware agnostic, various grasping technologies were reviewed for applicability and effectiveness. From the chosen grasping technology, a subset of robotic grippers that have been previously in use at nuclear facilities for various tasks were further reviewed to choose the best platform for testing and training purposes.

The first widely used grasping technology did not use the typical definition of grasping, and instead uses suction to secure an object. There are many pros and cons associated with this technology. Using a vacuum, or a region of low pressure density, to hold an object has been in use for many years. Many can even look in their kitchen drawer to find a turkey baster that uses the same principle to hold gravy, and have even employed the physics involved to hold an ice cube at the end of a drinking straw. If a force larger than the suction force knocks the object loose, or moves the object perpendicular to the direction of suction force, suction can be lost immediately, dropping the object. Vacuum grippers require a smooth, pliable suction-cup and a non-deforming surface to apply a force on to lift an object. Vacuum systems rely on pressurized air which require a pump or pressurized air. Another disadvantage of vacuum grippers is the vacuum mechanism itself. Because many vacuum

systems require a minimum air pressure differential, there exists a possibility that dust, dirt, and debris clog the system, which can be difficult to clean out. Some advantages are the small size needed at the end of an end-effector (Figure 1.3), the ability to pick up flat objects easily, and the quick action of lifting and dropping items. The data that can be recovered from this type of system is generally small, only having the ability to record air pressure and relative angle of the suction cup.

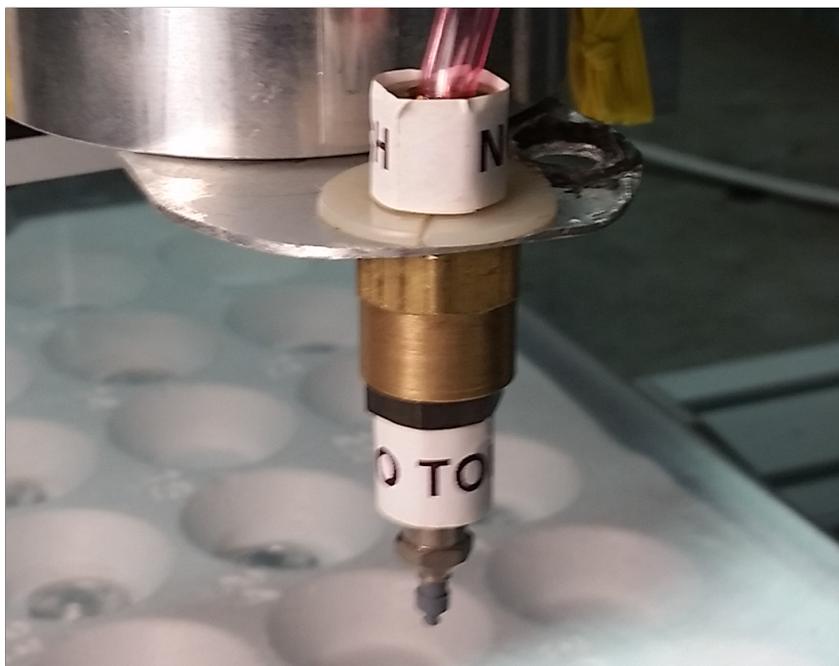
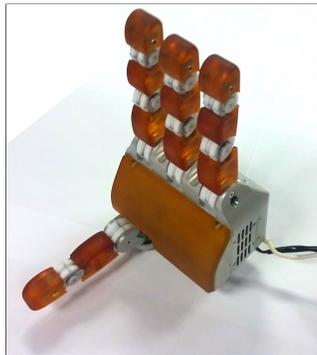


Figure 1.3: A vacuum system utilized in moving small cups. The gray portion at the bottom is the actual suction cup.

A competing technology to vacuum grippers involves the traditional clamping and encompassing grippers. These two technologies are the basis for thousands of different grasping systems, but just like vacuum systems,

they have their own advantages and disadvantages. Grippers in this field can be both large and small based on the technology that moves the grasping appendages. If pneumatic means are used to grasp an object, similarly to vacuum systems, their size can be extremely small, and they reduce the need to have a smooth surface on which to grab. If electronic means are used to move the grasping appendages, then the size is necessarily larger, as electric energy needs to be converted into a mechanical force on the appendages and onto the object. Encoders can be used on both systems to record torque, joint positions, and other facets of the system to generate a robust data log. Due to this capability, applying a learning algorithm is more applicable for clamping and encompassing grippers.



(a) Meka Hand



(b) Robotiq 2-Finger



(c) Robotiq 3-Finger

Figure 1.4: Images of potential robotic grippers.

Three grippers were evaluated for use in this effort. The first robotic hand reviewed is the Meka Compliant Hand [51] shown in Figure 1.4a. The Meka hand is a five degree-of-freedom, cable-driven, anthropomorphic hand. The hand is composed of four fingers each made from series elastic actuators.

It has payload of 2 kg and is made of aluminum and plastic which are not radiation or chemically resistant materials.

The second is the Robotiq 2-Finger adaptive robot gripper shown in Figure 1.4b [40]. The 2-Finger gripper is an actuator driven, parallel finger gripper. The mechanical linkages adapt to the shape of the object grasped, allowing secure grasping of cylindrical-shaped objects. This gripper can lift 2.5 and 5 kg using the 140mm and 85mm versions respectively.

Finally, the third is a Robotiq 3-Finger (Figure 1.4c). The adaptive robot gripper has the advantages of adaptive mechanical linkages, as well as more versatility than a 2-DOF "pinch" type gripper. The payload is 5 kg. Both Robotiq grippers are made of stainless steel, a radiation and chemically resistant metal, which is necessary in a glovebox that can contain radioactive and corrosive materials. The large size of the Robotiq can be a hindrance in glovebox usage, but the ability to handle tools similarly to human hands is a balancing benefit if it can execute the necessary tasks in the confined space.

After a more in-depth review of the gripper was accomplished, the Robotiq 3-Finger adaptive robot gripper was selected for use as a part of this effort for the reasons listed below:

- Availability
- Compatibility with tasks related to the funded research
 1. High (5 kg) payload
 2. Sufficiently (if not ideally) compact

- Compatibility with the nuclear environment related to the funded research
 1. Built from compatible materials
 2. Motor and electronics located away from the fingers which reduces potential radiation issues.
- Multiple grasping modes encourages configuration agnostic software development
- Low power
- Includes fail-safe mechanism
- Compatible with commercially available pressure sensors
- Robot Operating System (ROS) drivers exist
- Robot Operating System (ROS) URDF files for future collision detection motion planners previously developed by the NRG
- Installed sensors sufficient to know configuration state of the gripper

In addition to the above reasons, the Robotiq sensor already has some grasp validation capability. The 3-Finger gripper has multiple embedded sensors that can be utilized to detect objects. Current for the actuators can be read, as well as the estimated position of the fingertip. Using this data, Robotiq software guesses if it holds an object or not. The object detection sometimes does not work correctly due to the adaptable fingers of the Robotiq and their inherent flexing abilities that are not controlled by motors.

To increase the sensing capabilities for the 3-Finger, TakkTile Sensors

shown in Figure 1.5 have been attached to the Robotiq gripper. An individual Takktile sensor is composed of a barometric sensor that has been modified to sense touch (pressure, strain, curvature, or shear) and temperature through the embedding of the sensor in rubber [49]. This method increases the robustness of a touch sensor by removing the actual sensor from the objects with an intermediary material while maintaining a high pressure sensitivity from the barometric sensor.

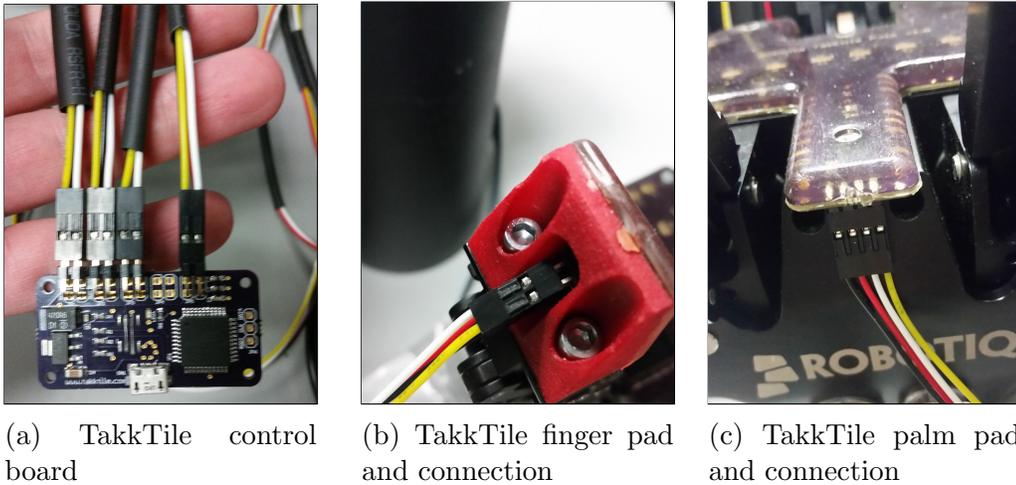


Figure 1.5: Takktile Robotiq components.

1.3 Grasp Validation

Grasp validation is typically done in an ad-hoc way, with each laboratory or company developing a particular method for each combination of gripper and task application. The proposed learning-technique will insert a grasp validation step after a grasp has been attempted. For example, the

Nuclear and Applied Robotics Group (NRG) lab at the University of Texas at Austin currently uses a multi-step method for determining and validating grasps. First, a heuristic grasp plan is generated based 3-D models of the object to be grasped and a vision system that accurately pinpoints the object’s pose (3-D orientation and position). These grasps are then ordered based on the closeness of the grasp point to the end effector of the robot. These grasps are planned utilizing multiple processing threads in *MoveIt!*, with the first successful plan executed.

Second, after the grasp plan has been executed, a two-step verification is checked simultaneously to ascertain whether the object has been successfully grasped. The Robotiq has a software check on whether the gripper thinks it is holding an object between its fingers, while the arm is raised a sufficient amount to obtain a reading from an attached force torque sensor. While this approach gives some increased confidence that the item is grasped, neither method explicitly determines the grasp quality.

The approaches developed as a part of this effort can be added before or after the second step of the NRG’s grasp planner or added independently in other planning frameworks.

1.4 Summary of Objectives

The objective of this thesis is to develop and evaluate the effectiveness of learning algorithms to automate the grasping and grasp validation tasks inside a glovebox. While the proposed algorithm will leverage some aspects

of the glovebox domain to achieve this, the results should be applicable to other domains with similar environmental constraints (i.e. low number of parts, rigid parts, constrained space, etc.) and the evaluation of the proposed solution should allow for the determination of the computation limits of the algorithm as it relates to these constraints. To achieve this, an understanding of the objects in the environment and the environment itself is needed. After a grasp has been executed by a planning framework, the learning algorithm will collect data from the embedded pressure sensors as well as sensors integrated in the gripper and quantify the quality of the grasp. Methods to complete the aspects of a grasp plan and execution exist. Validation is often the critical missing element for systems handling a general set of fragile, dangerous, or expensive objects.

1.5 Organization

An important corollary of achieving this objective is the evaluation and documentation of the challenges for using learning algorithms in the glovebox and (more generally) nuclear domain. Chapter 2 reviews literature on robotic grasp validation and learning algorithms. Chapter 3 presents the proposed learning solution for grasp quality quantification. The implementation and demonstration of the algorithm is shown in Chapter 4 along with an analysis of its efficacy. Chapter 5 encompasses a summary of efforts within this paper along with an outline of possible future work.

Chapter 2

Literature Review

Robotic grippers have been in use since the advent of industrial robotics. As technology improved, so have the grasping mechanisms and sensing technologies integrated within these grippers. The ability to leverage these sensors, however, has not advanced as fast as the technology. Compounding the issue of integrating new sensing capabilities in grippers into planning systems is the number of objects that a robotic gripper may grasp is - for all practical purposes - infinite. Normally, if the number of potential objects to be gripped is too large, machine learning approaches become intractable. Reducing or separating the types of objects a gripper will encounter on a task-by-task basis eliminates this intractability when combined with proper learning algorithm selection. To understand and decide on a method to integrate the specific technologies chosen for this task, a review of general methods of grasp validation, grasp planning, and probabilistic learning methods are covered below.

A general review of current grasp planning methods is presented in the first section. While not the focus of this effort, grasp planning methods, and the ideas behind each different method, impact the foundation of the type of grasp executed and scored. Following the review of grasp planners, grasp vali-

dation is dissected into three distinct sections that constitutes different aspects that are used within grasp validation architectures. The first section covers grasp planning, the precursor to grasp execution. The second section serves as an introduction to probabilistic methods and learning approaches that can be used in grasp verification, while the final part will describe specific modern methods for machine learning or probabilistic methods for determining grasp quality. These three parts span historical to modern methods of determining grasp quality, finishing with recent advances in literature.

The supervised learning algorithm applied later in this thesis is based largely on previous work within the field of machine learning. This chapter discusses previous work in various fields of robotics and machine learning that are relevant to grasp quality calculation. This is a non-comprehensive review of several approaches and supporting technologies for grasp validation.

Grasp validation is composed of two complementary concepts: grasp success and grasp quality. A grasp planner succeeds if the final executed grasp matches the planned grasp. Grasp quality determines the effectiveness of the executed grasp in keeping the object within the gripper that is holding it. This metric differs slightly compared to other efforts in quantifying the metric. For instance, Nancy Pollard [36] states "A grasp quality measure is an estimate of the suitability of a grasp for the task to be performed." The similarity between both our statements lies in the words "suitability" and "effectiveness". The metric used in this thesis is not as broad, focusing on one application; one in which grasping securely is paramount.

2.1 Grasp Planners

Grasp planners provide a set of directions to follow which will result in a grasped object. Usually grasp planners are software based which take information from the kinematics of a robot and sensors to plan a grasp. The number and variety of grasp planners are numerous, but a few core grasp planning methods are covered within this section.

2.1.1 Early Grasping Work

Early grasping work was limited in two ways: hardware variety and computational power. The first robotic arm used for research purposes was the Stanford arm developed in 1969 - a fully controllable manipulator capable of moving with 6 degrees of freedom. The Stanford arm included a parallel gripper - a simple two finger gripper (rather than suction, adhesion, or fixture-based grippers) where the finger's relative distance apart is controlled to grab objects.

Following parallel grippers, early grasping research focused on utilizing these grippers in "squeeze" or "pinch" grasps. These grasps are simple - with an object between the two fingers, close the fingers with enough force to hold the object. This research effort continued from 1969 to the late 1970s, until the invention of increasingly complex grippers and planning methods due to the advancement of computer chips. Early work included touch-based reaction sequences that relied on human-input to know where objects were located in order to grasp them [34]. Before vision systems and microchips were inte-

grated, many grasp planning methods followed the example sequence below:

1. Receive input of object location
2. Calculate torques necessary for movement
3. Move towards item position
4. Did left or right finger touch?
 - (a) If yes, move towards touch then close
 - (b) If no, move towards item position again

With recent hardware, pinch and squeeze planners have evolved to include various high-throughput sensors to calculate optimal grasp points on objects for parallel-grippers [9] [5] [48]. These grasp planners range from the example above to planners that include 3D visual systems to find features, such as anti-podal points on an object, to assure a solid pinch grasp. Many efforts before 2000 rely on *a priori* information on objects and their locations and use that knowledge to generate grasp points. For instance, Brost [5] calculates contact pairs for grasp points on objects, but relies on knowledge on the shape of the objects and the orientation of them as well. More modern efforts in "pinch" grasping involve knowing very little about the environment and location and instead gathering and making sense of information gained from sensors [48]. Andreas ten Pas and Robert Platt use 3D point clouds to recognize objects, locate good pinch grasp locations, and execute grasps autonomously with little or no information from the user.

In the late 1970s advancements in gripping technology included the first angled two-fingered grippers. The difference between these parallel grippers lie

in the motion of the fingers. While parallel technology kept the fingers parallel to each other, angled grippers instead actuate with a rotary joint at the base of the finger. To visualize an angled gripper, make a V-sign with your index and middle finger and bring them together. The design of these grippers could be argued to have been spurred forward based on uncertainties related to grasping. A much wider grasping area is available with a much smaller footprint than parallel grippers. Grasp planners evolved to work with the new grippers and are called Caging or Push-Grasping methods [37] [38] [12]. Push grasping is popularly used for tasks with large uncertainty inherent in the environment or the robot itself. Caging methods are primarily used when the object properties are not well known or manipulation of the object's orientation is needed and focuses on encompassing an object on all sides [53] [54] [14]. Caging methods at first were designed mainly to stop objects from escaping a grasp. This design principle focused on custom fingers specifically for the objects that needed to be grabbed to ensure caging grasps were possible. The basics remain similar to pinch grasping, but instead of finding antipodal locations, planners try to find a combination of points that restrict the movements of the object to zero when the grasp is correctly calculated. Recently, the idea of caging and the introduction of faster processing has combined with mobile robots and other advanced manipulation tasks [54] [14].

Push grasps provide a solution to the problem of uncertainty and has remained a popular method of grasping even today, moving from object location uncertainty to grasping within cluttered environments [6] [15]. Push

grasps rely on having a wide area in-between gripper fingers, or a large grasping area, that anything inside of will be grasped. Two methods are used to ensure a successful grasp: the geometry of the fingers are created in such a way to push objects encountered towards the center of the gripper or designed so that when the object touches the gripper at any point within the grasp area, the grasp should be successful. Push grasping itself is a simple process with minimal computation needed; it only needs a close location of the object and a collision-free trajectory towards the object.

After the two finger grippers, multiple finger systems were developed, though three and five fingered grippers are the most common. To fix an object in space at least 3 points of contact are needed. A human hand has 5 fingers, combined with minimizing the complexity of the human hand lead to the joint popularity of two main types of grippers in research and manufacturing fields. The closer a grasp comes to human functionality, the more models researchers can make between human grasp types and robotic grasp types. Many times these models involve a model of human grasps, that the robotic grippers can mimic. Figure 2.1 shows Cutkoskys taxonomy of prehension arranged by similarity [10]. This taxonomy is not exhaustive, but covers the most common grasps that can be used as a baseline reference for grasp analysis. Ideally a model-based grasp generator would be able to find an optimal grasp that would not fail; however, due to insufficient information, models, and computational time, many methods utilize reducing assumptions when generating potential grasps [21] [30].

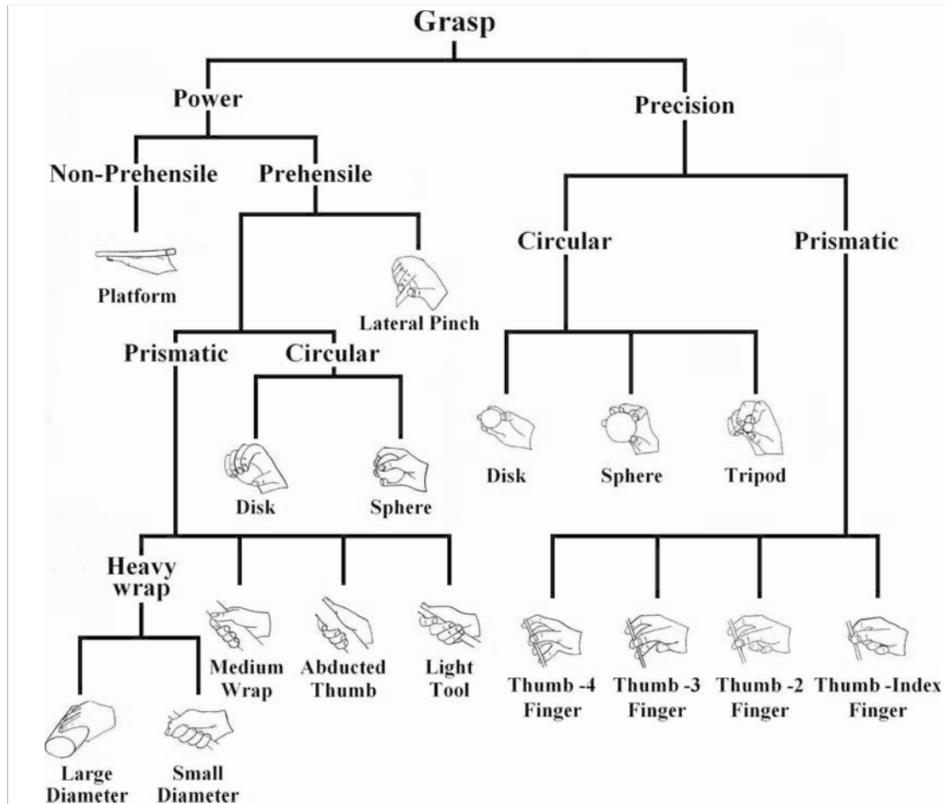


Figure 2.1: Cutkoskys taxonomy of prehension [10]

Some primary grasp planning methods have been explained or detailed above, but a key component of grasp planners used today, such as vision and 3D vision. Vision has played a key role in automating many grasping applications [48] [14] [15] [32] [30] [22] [43]. Incorporating visual data into grasp planners relaxes the requirements for input data on grasp planners but infers the need for object recognition to identify the now missing information [2]. The information processed includes the object name, the size, location, orientation, and even color in relation to the serial manipulator and gripper combination.

Haf_grasping [17] and agile_grasp [48] are recent efforts to utilize just visual data to calculate anti-podal and other suitable grasp locations on objects.

2.1.2 Other Modern Grasp Efforts

The previously discussed grasp planners are simple in nature, and have not truly explored the advantages that modern processing power and visual data can bring to grasp planning. These planners range from decoding visual data into primary shapes for grasp planners to machine-learning based approaches.

The easiest way to understand grasp planning from recent modern efforts is called "Primitive Shapes" [4] [32]. These range of planners involve taking in visual data, or physical models, and deconstructing the complex shape into its primary constituents. To make this easier, Brabec [4] assumes the number of objects that a gripper may need to grasp is usually limited by the scope of its working conditions. Brabec focuses only on objects that may be encountered at the proposed deployment location of the robot. An example of shape primitive deconstruction can be seen in Figure 2.2. The number of locations, orientations, and grasps create a continuous space on any object that is presented, but by reducing the object set and focusing on pre-determined styles of grasps, the problem becomes tractable using a simple heuristic approach.

The objects are first visualized and reduced to their shape primitives. Brabec uses a numerical model-based approach to retrieve viable grasps from

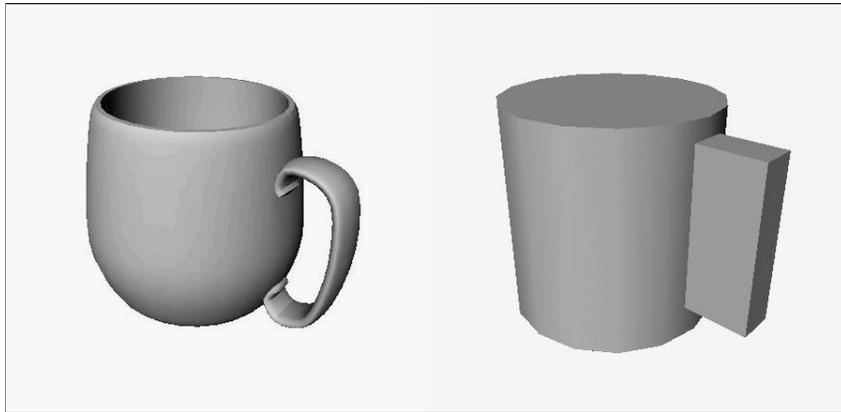


Figure 2.2: A primitive shape decomposition of a coffee mug [4]

a preplanned set for a given primitive. In simulation, the grasps are reduced to reachable and non-reachable ones, before finally selecting the closest grasp. Shape primitives are a perfect example of why grasp verification is needed: by employing such a simplifying grasp planner, the resulting grasp may be unsteady due to the reduction in complexity of the geometries and can potentially leave out valuable information. For instance, in Figure 2.2, the handle may be selected for a pinch grasp, but the hole in the handle has not been modelled which could lead to future problems.

Primitive shapes uses the geometry of the robot gripper and the object to generate known good grasps on simple shapes, but using the complex shapes and calculating the forces between the gripper and the object is much more precise. These grasp planners also function as physics models of the environment and the object and can be found in many areas of current research and are constantly evolving [44] [45] [18] [27]. The requirements to use these systems are much higher than the grasp planners listed before due to

the exacting nature of physical models and simulations.

Some efforts have involved the combination of machine learning and grasping simulators. Pelosof et. al. [35] attempts to solve the approach of finding good, or close to optimal grasps, using a combination of numerical methods and machine learning methods to interpolate the object's surface, commonly referred to as a support vector machine (SVM) [35]. Pelosof assumes the stability of the grasps through simulating the grasps in a physics simulator GraspIt! [18]. After potential grasps are generated and sorted based on the results of training the SVM classifier with the results from GraspIt!, the grasp is then executed. Generated grasps also assume that the robotic gripper's palm should be parallel to the surface of the object to reduce the state-space of generated grasps. This method shortcuts grasp evaluation in simulation by basing the potential grasps on previous results. This can be dangerous if the materials are not rigid, or if the surface of the object does not conform to the assumptions made when classifying the grasps. An example can be seen in Figure 2.3, the optimal grasps found through SVM classification involve very little contact between the gripper fingers and the object, if the object deforms or does not follow the characteristics provided the grasp might fail.

Grasp planners don't quantify grasp verification, but may deduce a grasp quality before the grasp is executed. Often, industrial solutions combine grasp planners with tactile or other sensors to check if the gripper contains an object, but do not directly calculate grasp quality. Important details can be

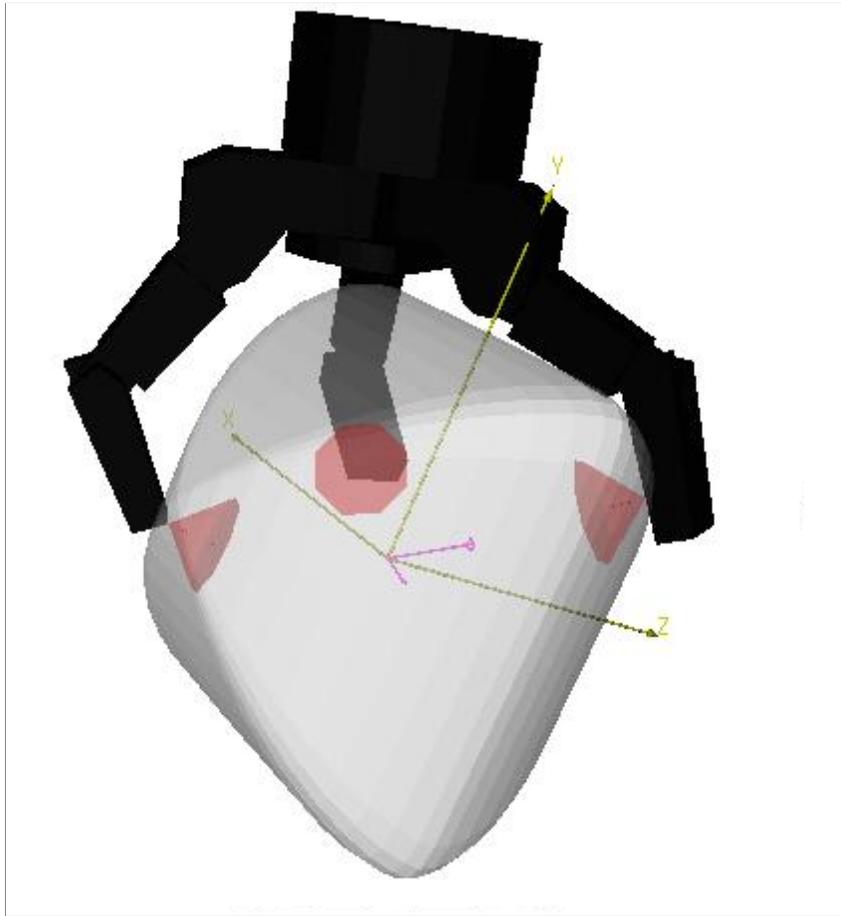


Figure 2.3: An SVM sorted optimal grasp generated through GraspIt! [35]

missed, including the grasp viability during dynamic motions, contact tasks, or EEF rotations. While not perfect for determining grasp quality after uncertainty inherent in grasp execution, these methods provide a critical service by generating the best grasp possible with limited information. These efforts address that lack of information after a grasp has been executed to ensure the upcoming task can be completed without a grasp failure. Some methods may not work well with this secondary verification, such as Pelosof's approach

above, as their assumptions do not allow much contact between potential tactile sensor locations and the object grasped.

2.2 Non-Learning Based Grasp Validation

Grasp quality determines the effectiveness of the executed grasp in keeping the object within the gripper for a given task. Many different sensors, and combinations thereof, have been utilized to quantify grasp quality. The sensors range from vision systems, joint torque or joint current sensing, and force torque sensors embedded on the end effector of the robot or in the gripper hardware itself.

The earliest grasp validation efforts involved watching joint currents, or their calculated joint torques, in serial manipulators to determine whether an object is being lifted by the arm. This method provides only an indirect validation, as at most, without swinging the object to determine its kinematics, it is impossible to determine the quality of the grasp itself. While initially not very useful, researchers, and companies, have also applied these joint-monitoring technologies to other areas than just the manipulators and have instead added them to the fingers on the grippers, which gives very useful force feedback from the object being gripped. These applications range from surgical robots, prosthetics, and fruit sorting [16] [29] [25] [20]. Another extension, on robots that are not conducive for this type of monitoring (closed-source, proprietary, older), are force/torque sensors installed on the end-effector of a robot in-between the gripper and the manipulator. The force/torque sensor

then enables monitoring of the weight/force of grasped objects.

Many current grasp planning efforts utilize vision to localize objects and determine their properties [2] [47] [8]. Including a visual grasp validation tool requires no new robotics or gripper hardware, though the object may need to be brought closer to the vision system for verification. One drawback observed in the NRG lab was that adding visual data can be resource intensive and even be detrimental if the gripper and object are obscured. Most information needed to determine the quality of a grasp can be determined through the gripper itself and various sensors that are less resource intensive. To address this issue, vision systems can be utilized only briefly to locate and orient objects in virtual space and then disabled until more information is necessary.

Hebert et al. developed a method to integrate these three approaches: vision, joint torque, and force/torque sensors into a method for estimating the location and orientation of an object held by a manipulator gripper [19]. The hybrid system employed in the paper resulted in a robust system that can tolerate failures of one or more systems, and helps prove that a combination of techniques, when used together, can improve grasp planning and grasp validation.

2.3 Introduction to Probabilistic Methods in Robotics

Tomorrow's application domains differ from yesterday's, such as manipulators in assembly lines that carry out the identical task day-in day-out. The most striking characteristic of the new robot

systems is that they operate in increasingly unstructured environments, environments that are inherently unpredictable. An assembly line is orders of magnitude more predictable and controllable than a private home. As a result, robotics is moving into areas where sensor input becomes increasingly important, and where robot software has to be robust enough to cope with a range of situations often too many to anticipate them all. Robotics, thus, is increasingly becoming a software science, where the goal is to develop robust software that enables robots to withstand the numerous challenges arising in unstructured and dynamic environments.

...

Probabilistic robotics is a new approach to robotics that pays tribute to the uncertainty in robot perception and action. The key idea of probabilistic robotics is to represent uncertainty explicitly, using the calculus of probability theory. Put differently, instead of relying on a single best guess as to what might be the case in the world, probabilistic algorithms represent information by probability distributions over a whole space of possible hypotheses.

-Probabilistic Robotics by Thrun, Burgard, and Fox [52]

The above quote from Probabilistic Robotics by Thrun, Burgard, and Fox [52] provides a good description about why robotics benefit from the inclusion of probabilistic methods (also known as machine learning) in addition to covering in great detail machine learning as a whole. There are several methods described below that cover some of the most commonly employed machine learning methods in robotics today and a brief introduction of what

is involved with a machine learning task.

2.3.1 Supervised Learning

Supervised learning attempts to learn from a set of labeled examples provided by an expert. [46] For this research, supervised learning would consist of a data-set that contains the sensory readings described in Table 3.1. The expert's label, consisting of a score within $[-1,1]$, is used to train the system to classify past and present grasps. This approach relies heavily upon the expert's feedback, as supervised learning algorithms expect for the expert's feedback to be optimal and correct. Neural networks, decision trees, perceptrons, ensembles, and many more methods can be used to accomplish supervised learning, with each having their benefits and detriments. This effort uses a perceptron that learns parameters in a scoring function.

2.3.2 Reinforcement Learning

Reinforcement learning is composed of many different methods and frameworks that offers roboticists a set of tools to design and solve difficult engineering problems. Reinforcement learning (RL) allows a robot to find an optimal solution to a problem over time by utilizing trial and error. Through trial and error RL attempts to map situations to actions to maximize a reward signal designed by the user or the situation [26]. Exploration and exploitation are key points for reinforcement learning, as exploration attempts to explore the state-space and exploitation tries to take advantage of promising actions.

Inverse reinforcement learning (IRL), also known as apprenticeship learning, is a subsection of reinforcement learning that does not have a known reward function, and instead tries to extract a reward function given observed, optimal behavior. By observing an expert demonstration that is attempting to maximize a reward function, IRL tries to apply RL to maximize the "guessed" reward function, with many times assuming the reward function is expressible as a linear combination of extrapolated, or previously given, features [1].

2.3.3 Fuzzy Logic

Fuzzy logic is a form of soft computing; an approach to constructing computationally intelligent systems by combining many fields of expertise. Fuzzy logic is normally used in conjunction with neural networks to give a result that is easy to understand. For example, in this paper an arbitrary scoring function that outputs a real number between -1 and 1 scores the executed grasp. If this result is given to a human to interpret, the interpretation would vary between different users. Fuzzy logic captures this variation and quantifies how the user or expert would classify specific ranges of scores, based on the input, as good, bad, ok, etc. [23]

2.3.4 Machine Learning Components

Most machine learning algorithms share a core set of common components or ideas that lay the foundation to learn from uncertainty. These are listed below in no particular order.

1. Belief
2. Prior
3. Training Data
4. State Space
5. Features

The *belief* represents what the current system knows, or thinks it knows. A *prior* is the belief state of a new, or completely untrained system with uniformly distributed initialization when the system has an equal chance of choosing any response. This belief is modified through giving it a set of *training data* that contains, in the case of supervised learning, an input object and desired output. A single input object represents a state, or set of data values that represent a distinct configuration or activation of sensors. An example of a state would be 'ON' if the object in question was a light switch and it was in the on position. A more complex state would be a set of data values for all light switches in a house, and all the possible combinations of these values is called the *state space*. Finally *features* can be considered a single variable or groups of variables that are used for machine learning. In the previous example, each light switch is a feature that can be learned from, but if included with all of the variables in a home, the set of light switches could be defined as one feature.

2.4 Efforts Involving Learning and Grasp Validation

From the literature surveyed above, it is clear that several grasp planners exist in the literature and the robots/grippers proposed are suitable for use in a glovebox. But there is a lack of viable grasp validation strategies that ensure items grasped will not be dropped when lifted without specifically defining the object set and/or employing a method that is robust enough for hazardous environment. Some approaches, covered next, attempt to combine machine learning with grasp validation metrics in order to bridge this gap, but for different environments.

A recent approach for validating grasp configurations was researched by John et. al. [24]. The researches attempt to specify ranges over which a grasp is valid from examples submitted from participants online. An example, in Figure 2.4 shows some example good grasp configurations given by users online. This effort had both positive and negative results, as the effort involved complex action executions that are prone to generating complications. John et. al. used various contact points and mechanical sensors to generate future potential grasps from the given data sets by participants. While this effort can be used if your objects are commonly available or recognizable, but when objects cannot be released for viewing, aren't recognizable, or differ from common items by obvious or hidden traits, this approach falls short.

Another approach by Romano et. al. [41] attempts to selectively change applied force based on the type of object grasped and tactile feedback. This approach trades-off damage done to grasped objects and reliability of keeping

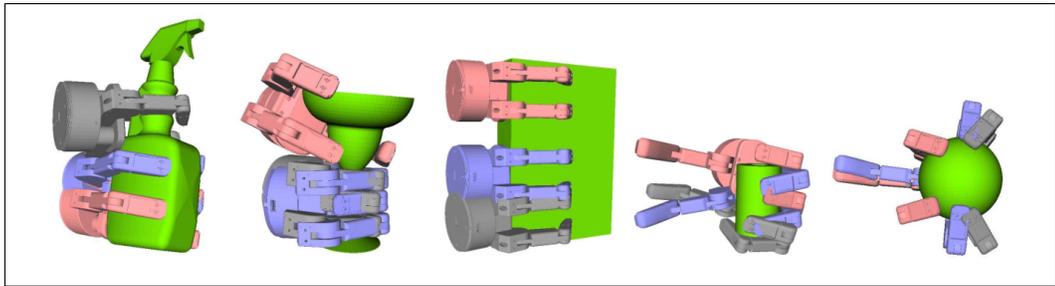


Figure 2.4: Five objects with three different grasps given by participants on object models [24].

the object within the gripper as a trade-off. The gripper will maintain or attempt to maintain a grasp if changes are detected, however the applicability of this approach is limited to objects that cannot be damaged through force-increases if an object starts to slip from the grasp. This trait is undesirable in many applications but can help a situation if the final grasp is unstable; a state this paper attempts to correct or warn before it can happen.

2.5 Summary

Early grasp selection and validations have been focused on model-based methods that do not fully employ the capabilities inherent to tactile-sensing robotic grippers for grasp validation. Many papers do not deal specifically with utilizing tactile sensors on the gripper to validate grasps, and instead focus on generating valid grasps or saving bad grasps, all of which compliment the aim of this paper. With the efforts of other authors encompassing grasp generation and manipulator motion, there exists a gap for a needed validation step. To accurately validate stable grasps, an online, or changeable machine-learning

technique is vital to integrate

Chapter 3

Learning Solution and Data Gathering

This chapter covers details on the implementation of an Inverse Reinforcement Learning Algorithm (IRLA). *Inverse Reinforcement Learning* represents the idea that the user has an unknown reward function associated with a group of features the algorithm will attempt to learn. The IRLA algorithm is designed so it fits the task at hand while retaining the applicability to other domains and sensor clusters. The algorithm is hardware agnostic, though it was implemented for applications in a nuclear manufacturing environment. First the algorithm itself, as introduced by Abbeel et al. [1], will be presented followed by the modifications and implementation details that cover the more application specific qualities of the completed experiments.

3.1 Iterative Improvement

The foundation of the IRLA algorithm lies in the iterative improvement method utilized in *Learning Trajectory Preferences for Manipulators via Iterative Improvement* by Abbeel et al [1]. The algorithm contains two basic components that allow learning to take place: a Trajectory Preference Perceptron (TPP shown in Figure 3.1) and the scoring equation the perceptron updates.

The original goal of this combination was to learn safe trajectories via users giving feedback on executed trajectories of a serial manipulator. The TPP updates the weights associated to various features by increasing the weights that help the scoring function reach the desired answer while decreasing the weights that act adversely to finding the correct answer. The application of the perceptron differs from a supervised learning algorithm as there is not a clear, definite separation from the sensor responses and the expert’s feedback that the specific grasp is bad.

```

Initialize  $w_O^{(1)} \leftarrow 0, w_E^{(1)} \leftarrow 0$ 
for  $t = 1$  to  $T$  do
  Sample trajectories  $\{y^{(1)}, \dots, y^{(n)}\}$ 
   $y_t = \operatorname{argmax}_y s(x_t, y; w_O^{(t)}, w_E^{(t)})$ 
  Obtain user feedback  $\bar{y}_t$ 
   $w_O^{(t+1)} \leftarrow w_O^{(t)} + \phi_O(x_t, \bar{y}_t) - \phi_O(x_t, y_t)$ 
   $w_E^{(t+1)} \leftarrow w_E^{(t)} + \phi_E(x_t, \bar{y}_t) - \phi_E(x_t, y_t)$ 
end for

```

Figure 3.1: Trajectory Preference Perceptron. w_O and w_E stand for object and environmental features.

The scoring equation itself is a linear combination of feature weights and their corresponding feature values. An example of this would be how much confidence does a person have that one feature’s values correspond to a good grasp; that is the feature weight and the feature value could be one particular sensor’s reading.

3.2 Inverse Reinforcement Learning Algorithm Design

IRLA uses a set of data points composed of sensor readings and user feedback to form a scoring function that scores future grasps based on the supplied data. IRLA requires a specific feature list for each application based on available sensors, but the underlying framework (preference perceptron and scoring function form) is constant throughout applications.

3.2.1 Features

Features can be any quantifiable variable associated with a grasp where IRLA can learn of a relationship between the feature and a successful grasp. Separating variables by feature, instead of lump-summing the values, allows a more precise measurement tool to identifying the most influential factors in a good grasp. High importance features should have a larger magnitude weight associated with the feature, though differentiating correlation and causation is difficult when more sensors are used. For instance, if two sensors read the same or near the same for every good grasp, which one is actually important? Do they hold equal importance? While a good track for future study, due to the application specificity of finding this distinction, it will not be progressed further. Carefully selecting the features of a learning algorithm is extremely pertinent to the quality of the results received, so pruning sensors based on perceived significance is not suggested.

The features chosen for this application are listed in Table 3.1. These features encompass tactile sensors, finger positions, hardware detection, object

type, and other settings on the Robotiq gripper. They are all of the currently available sensor readings available through software interfaces for the Robotiq plus a user executed grasp validation feedback feature.

Feature List		
Feature	Quantity	Range
Shape	1	{Cuboid, Amorphous, Flat, Cylindrical, Hemisphere}
Maximum Pressure	1	{0,1,...,244,255}
Pressure Sensors	36	{0,1,2,3}
Robotiq Object Detection	1	{-1, 1}
Robotiq Force Setting	1	{-1, 1}
Robotiq Finger Positions	3	{0,1,...,244,255}
Expert Feedback	1	{-1,0,1}
Total	44	-

Table 3.1: Features and their respective reporting range.

The number of learning features yields 44 different variables that the IRLA algorithm will attempt to match their values to the grasp quality. The maximum pressure recorded for each object allows for an indirect method to determine the relative frictional forces of different grasp configurations [41]. The Takktile sensors generate a large range of response for pressure readings, which increases the variance in the system if the data was learned from without modification. Binning the sensor readings into four bins based on the range from no pressure reading to the maximum pressure reading reduces the complexity of the feature. The 36 pressure sensors are scaled on the range listed below.

Pressure Range

- 0 (0 pressure to 10% of maximum pressure read)
- 1 (10% to 40% of maximum pressure read)
- 2 (40% to 70% of maximum pressure read)
- 3 (70% to 100% of maximum pressure read)

Figure 3.2 shows a representation of pressure sensor readings separated into the defined ranges. The Robotiq object detection is a true or false reading sent from the Robotiq and is either a -1 (no object detected) or 1 (object detected). The Robotiq force setting is recorded as either -1 or 1 based on whether 50% or 100% of the maximum force the Robotiq can use is used applied to the Robotiq. The final data collected from the sensors, and potentially the least revealing, is the final finger positions of the Robotiq. Normally this data can be leveraged with a grasp database and compared to known good grasp configurations [11] [3]; however, this approach it does not translate to a 3-finger, non-anthropomorphic gripper due to a lack of a grasp database and a set of universal tactile sensors on the Robotiq.

A final data point is added by an expert, a feedback on the perceived quality of the grasp. The expert manually applies a moderate force on angles directed outwards and inwards from the Robotiq palm in an attempt to mimic small bumps or jostling the object may undergo during transport. These forces are largely user-determined in the case of this experimental setup; however the same learning can be done using a different process. For example, moving the gripper and object in a predetermined path using a serial manipulator to

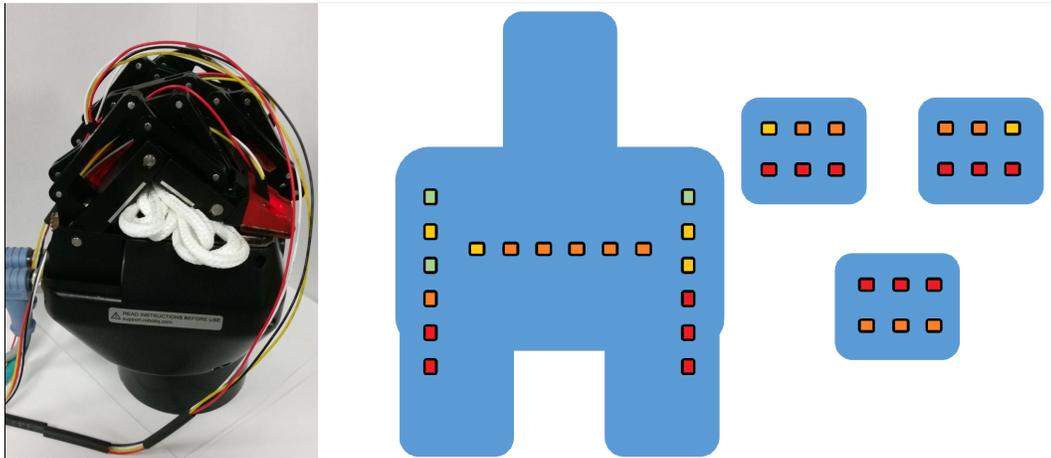


Figure 3.2: A Takktile sensor reading obtained from grasping a rope. The colors represent the range of forces normalized by the maximum force. Green represents 0 - 10% of the maximum force, Yellow 10% - 40%, Orange 40% - 70%, and Red 70% - 100%. (Green = 0, Yellow = 1, Orange = 2, Red = 3)

mimic movement forces or using the manipulator to bump objects with the same force each time. In this case a force was mimicked using a rubber mallet and a trained evaluator to apply a 20 N force each time. Larger forces were ignored to keep objects intact and from previous experimentation showing that after 20 N the Robotiq would go into a fault mode. While this testing measure does not account for all cases of impacts or potential object movement forces, it accounts for the most common cases of bumps or jostling in the NRG glovebox environment before a hardware fault would stop further motion. The expert grades the resulting grip on a scale of $\{-1, 0, 1\}$ based on the results of the test. A score of -1 is given if the object completely falls out of the grip, a score of 0 is given if slip or object movement is detected in the grip but the object remains in the gripper or if the object's grasp might pose an

a risk during further manipulation execution, or a score of 1 is given if the object remained firmly grasped by the gripper. The feedback is reliant upon the expert’s underlying definition of a hazardous grasp for scoring a 0 or 1. This feedback differs between tasks, object sets, and environments. Figure 3.4 shows a cylindrical object being grasped by the Robotiq gripper and Takktile sensors.

3.2.2 Scoring Function

The scoring algorithm chosen is $Score(x, y; w) = w \bullet \varphi(x, y)$ [22] used in *Learning Trajectory Preferences for Manipulators via Iterative Improvement* by Jain et al. The similarities between the use-case Jain et al. presents is similar in structure to the use-case and available features presented above in Section 3.2.1. First, the learning algorithm framework does not decide actions and only filters, or ranks given trajectories based on past experiences. A grasp quality analyzer seeks to rank the current grasp versus other types of grasps previously learned (not generate grasps) and uses a feedback method that improves the underlying algorithm.

$$Score(x, y; w) = w \bullet \varphi(x, y) \tag{3.1}$$

The Scoring function in 3.1 is composed of a weight vector, w , that will be learned, and $\varphi(\cdot)$ represents features that describe the current grasp. The x is a generic variable that describes the shape of the object, while y represents the available features received for the grasp for this context x .

The implementation of the scoring function Equation 3.1 differs from the one used by Jain by implementing a more rigorous testing mechanism rather than biased user preferences. The ranking of trajectories is instead replaced by scoring of grasps with a prompt for expert feedback.

3.2.3 Interaction Model

The Interaction Model discussed in *Learning Trajectory Preferences for Manipulators via Iterative Improvement* [22] is modified to better fit grasp quality training. The steps below repeat for each data gathering measure, or distinct grasp. This model can be instituted as an online algorithm so that as work continues in an environment, bad grasps while in production can still be used to train the learning algorithm.

Step 1: The gripper will close upon a given object placed within its movement envelope and record all of the available sensor data. The closing step is controlled by the user or an autonomous program that detects when a grasp is made (In this case user controlled).

Step 2: The program will receive a context x for the object grasped (the shape of the object). The model then uses the current known scoring function to grade the grasp. The user then will enter the correct determination for the grasp. These two values are compared: *guessed* versus *user* determination.

Step 3: The algorithm now updates the w parameter of $Score(x, y; w)$ based on the expert feedback from step 2. The algorithm then loops back to

step 1 after releasing the object. The smaller the difference between the expert feedback and the scored response, the less the scoring function will change.

Regret: This aspect of the algorithm explained in Jain 2013 remains largely unchanged. The regret function is shown at equation 3.2. This function compares the calculated score y_t against the optimal feedback y_t^* to maximize the expert’s unknown Scoring function $s^*(x, y), y_t^* = \operatorname{argmax}_y s^*(x_t, y)$.

$$REG_T = \frac{1}{T} * \sum_{t=1}^T [s^*(x_t, y_t) - s^*(x_t, y_t^*)] \quad (3.2)$$

3.2.4 Software Application

Two applications were implemented and investigated using C++ and ROS to correctly gather, store, and update a database that contains the learned weight vector and scoring function. One application operates the Robotiq and collects sensor data. The second application updates the scoring function and keeps track of regret described in the previous section. A basic representation of the application and interaction structure is seen in Figure 3.3.

The first application, named **Mechanical Control**, is designed to respond to commands from **Scoring Control**. **Mechanical Control** interacts with and controls the Robotiq through two ROS nodes, a communication node developed by Robotiq and a controller node developed by the NRG. **Mechanical Control** also records data from the Takatile sensors via another

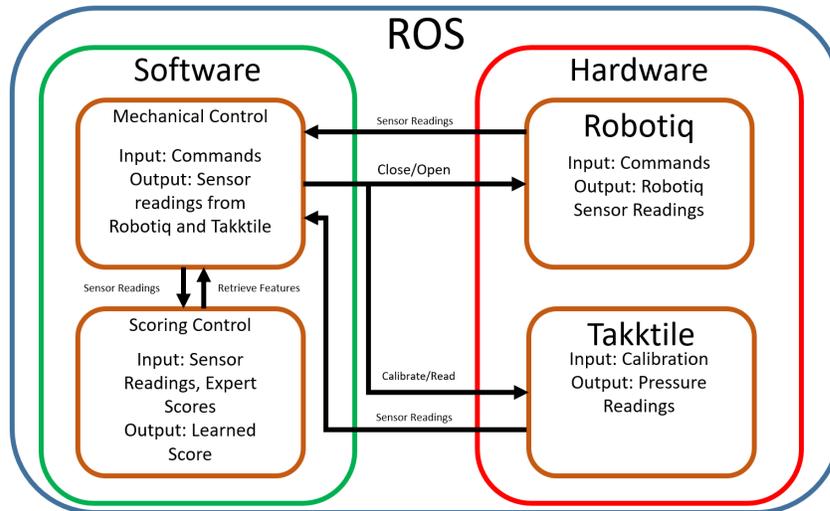


Figure 3.3: Representation of active nodes within ROS and their basic communication structure.

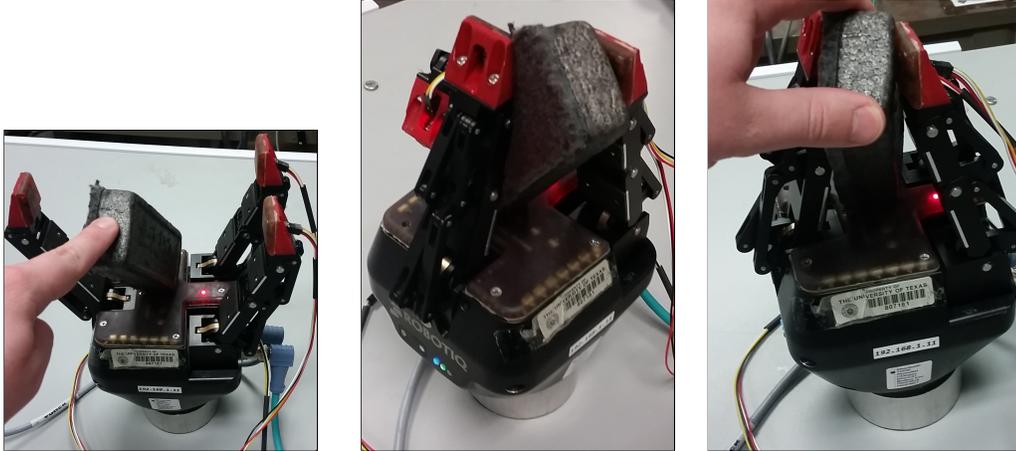
ROS node.

Scoring Control is the software framework for combining sensor data and communicating with **Mechanical Control** to time the release of an object and grasping an object. After an object has been grasped, **Mechanical Control** sends a response to **Scoring Control** containing all of the sensor data that it can see. After the different feature values are recorded, **Scoring Control** interacts with the user and receives input on the grasps' scores.

3.3 Data Gathering

The small number of objects allows for an in-depth generation of data points for each object instead of a breadth-search of many objects. The objects

selected in section 4.3 are run through a multitude of different configurations within the Robotiq gripper to ensure robustness.



(a) The Robotiq gripper in the Ready position with object held in place for the grasp.

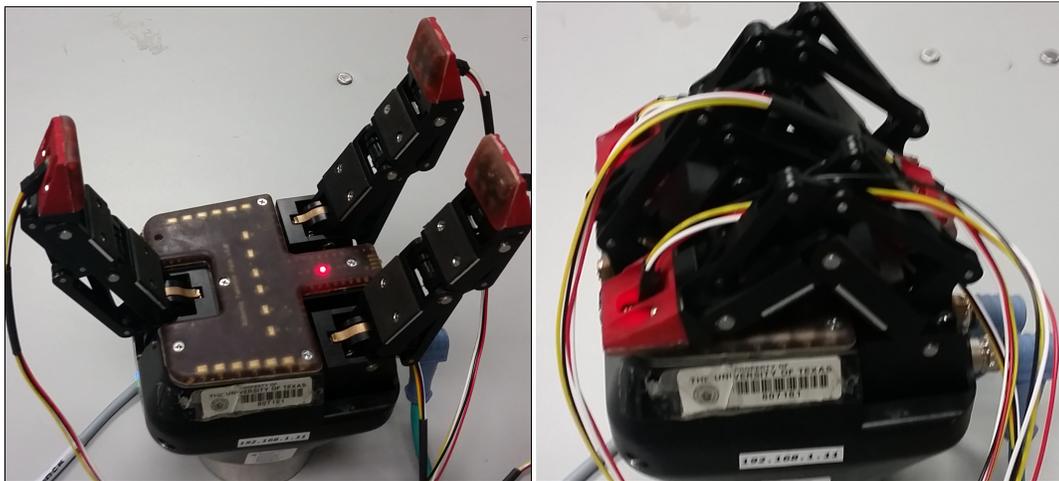
(b) The Robotiq has grasped the held object.

(c) The expert bumps and tugs on the object to assess the quality of the grasp.

Figure 3.4: Robotiq grasping data collection visualization

Initially the Robotiq and Takktile sensor combination are open in the "ready" position which is shown in Figure 3.5. The user is prompted by `Scoring Control` for the shape of the object to be trained and then waits for the user's next input to continue the learning process. The next steps are shown in Figure 3.4. Each test object is placed in novel positions within the gripper. When the object has been placed within the gripper, the user presses "Enter" to start the training. The Robotiq is commanded to "Close" to the position shown in Figure 3.5. `Mechanical Control` will randomly choose whether to use maximum or half-maximum force and will then attempt to

completely close the Robotiq’s fingers. After the Fingers on the Robotiq have reached the maximum force allowed by the current force setting. `Mechanical Control` waits for 1 second for the grasp to settle before sending a request for a sensor reading to the Takktile ROS topic and then retrieves the various Robotiq sensor values from Robotiq’s ROS topic. After receiving all of the available sensor information, `Mechanical Control` sends the available data back to `Scoring Control` to continue the training task. `Scoring Control` records the sensor data onto a file and waits for the expert to test the grasp robustness and score the grasp. This task is repeated for each object for 50 different grasp configurations per object, with 3 different objects in each shape classification.



(a) Robotiq in a basic (open/ready) mode

(b) Final commanded position, "close"

Figure 3.5: Robotiq basic movements and commanded positions

3.4 Summary

The initial algorithm is described as created and implemented by Abbeel [1] and Jain [22]. The learning algorithm is modified for the specific environment and learning goal of grasp validation. Learning features are identified and described to maximize sensor utilization. Final changes are made to the interaction model and the data gathering steps are detailed using the new interaction model.

Chapter 4

Demonstration and Analysis

4.1 Hardware

The Inverse Reinforcement Learning Algorithm - IRLA - is developed for the Robotiq 3-Finger Adaptive Robot Gripper and Takktile sensors in this research endeavor. The Robotiq 3-Finger Adaptive Gripper has the best mix of available features for use within a potentially radioactive or hazardous environment. The Robotiq was a popular selection by teams participating in the Defense Advanced Research Projects Agency's (DARPA) Grand Challenges [13] which was a contributing factor in the choice to work with the Robotiq. The final configuration of the gripper is shown in Figure 4.1 with the Takktile sensor package attached to the palm and fingers of the Robotiq. The gripper has its advantages and disadvantages detailed below in Table 4.1 that also contributed to the final gripper selection.

The Robotiq has four primary modes: pinch, basic, scissor, and wide mode. Pinch mode is designed for grasping parallel surfaces, scissor mode for tiny objects, basic mode for most other shape types with wide mode for objects that can't be grasped well by the other three modes. To simplify the learning process, only the most commonly used mode - basic - was chosen, so many

advantages of the Robotiq have not been leveraged to their full potential and certain objects and object types may be disadvantaged by this selection. The selection for basic mode was also chosen to utilize the tactile sensors on the palm pad more effectively as the pinch mode does not apply a perpendicular force to these sensors.

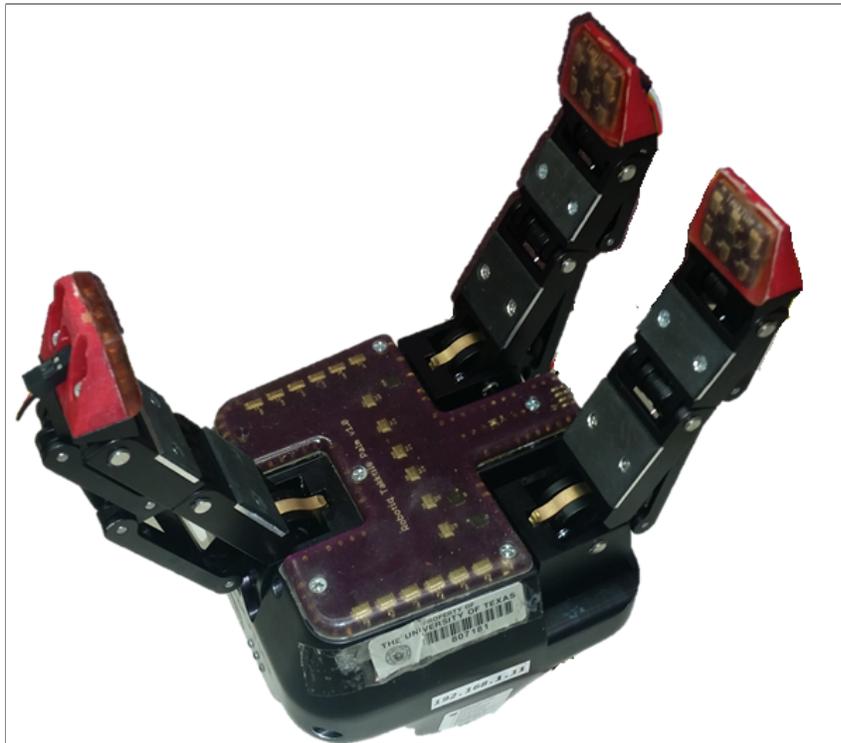


Figure 4.1: Final Robotiq 3-Finger Adaptive gripper and Takktile sensor suite.

4.1.1 Touch Sensors

IRLA was tested with the Takktile pressure sensor attachments shown in Figure 4.1. The Takktile sensors were chosen due to an existing ROS software package that allowed seamless software and hardware integration with the

Robotiq 3-Finger Adaptive Gripper	
Advantages	Disadvantages
Robust	Large, Bulky
Used in DARPA challenges	Non-anthropomorphic fingers
Working ROS packages	Adaptive in only one dimension
Standard-use in laboratory environment	Error associated with adaptive fingers
Moveit packages	No encoders on finger joints
Can grasp large range of objects	
Relatively high payload	
Fingers are mechanical (no electrical components extremely near radioactive material)	

Table 4.1: Robotiq table covering the various advantages and disadvantages lab members experienced by employing the Robotiq 3-Finger Adaptive gripper for various applications.

Robotiq. The sensors' size also fill the primary role of increasing the number of sensors available for learning grasp quality.

Takktile sensors are not the only tactile sensors that have been used to validate a grasp. There are many ways to sense a difference of pressure between two surfaces. Just detecting this difference ignores shear forces. The tactile gradient was also usually ignored. In a 2002 paper [28] Lashi uses a tactile array that consists of Force Sensing Resistor technology. The sensor array avoids shear forces, focusing instead on profiling the force spread and gradient of objects gripped in the humanoid hands.

Figure 4.2 describes a system in which a sensory subsystem detects and processes information from the environment. The subsystem takes into

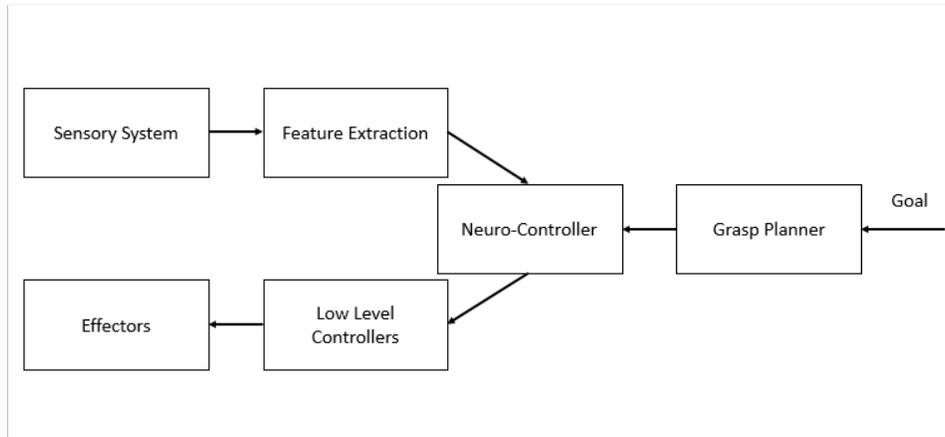


Figure 4.2: Function scheme presented in Laschi’s paper. [28]

consideration tactile feedback from the fingertips on their robotic hand from a ARTS tactile array. A system then uses that information, along with other pertinent information, to create a somatic grip model for the object the robotic hand is grasping.

Takktille sensors differ from other standard tactile packages in that Takktille re-purposes barometric sensors via injection molding a layer of rubber to convert touch pressure into barometric pressure. The ratio of tactile to barometric pressure can be customized to modify the force-reading profile shown in Figure 4.3 via differing rubber type and thickness. This feature allows Takktille sensors to be modified for many different applications that require differing sensitivities.

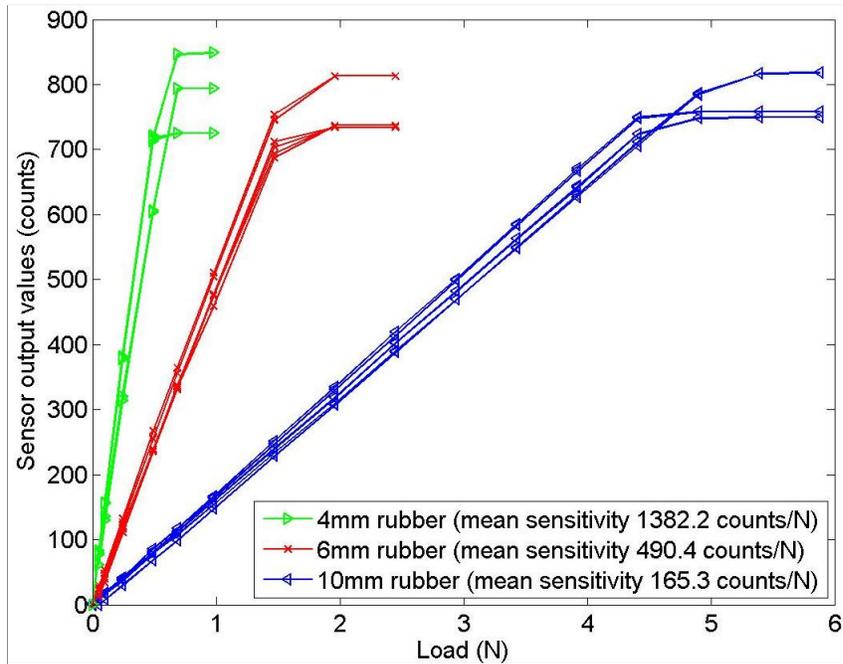


Figure 4.3: From the paper *Inexpensive and Easily Customized Tactile Array Sensors using MEMS Barometers Chips* [49]. Sensor output values versus applied surface load for differing rubber thickness.

4.1.2 Robotiq Hardware Sensing Capabilities

Based on if the fingers servo/motor encoder reaches its ordered to position, detailed within the Robotiq 3-Finger Adaptive Gripper Instruction Manual <http://support.robotiq.com/pages/viewpage.action?pageId=590045>, the built-in object detection uses an internal algorithm that determines whether the fingers may be in contact with an object. This algorithm is based on whether the fingers of the gripper reaches the final commanded position, and is only accurate if the object is larger than a few millimeters. The rope, in Figure 3.2, was not detected with this method. This result has been con-

firmed with other thin objects such as business cards, wooden pencils, various types of paper, and cardboard. The object detection is published to ROS from the gripper represented by a true/false declaration. The object detection also cannot detect a change in the object detection as once the Robotiq has finished closing, the declaration does not change until another command is sent to the gripper asking to move the fingers to a different position. Unlike the Takktile sensors, the object detection cannot be used continuously to check whether an object is still grasped. The underlying cause is from to the Robotiq fingers "locking" into position once the gripper has finished its move (the fingers cannot be reverse-driven or moved), allowing for limited amount of finger adjustments at this level.

Results are shown below for the correlation between the Robotiq object detection and different shapes in Table 4.2. Correlation is used in this context to represent the fraction of results that when the Robotiq detected an object that there was a good resultant grasp as determined by the user. There is a weak positive correlation between the object detection and the effective grasp quality for each type of object, but for amorphous and cuboid objects the results are favorable (better than random 50%). For cuboids, the flat, opposing faces lead to strong grasp points. For amorphous objects, even if an object starts to slip out, usually there is a strongly pinched location on the object preventing it from fully leaving the gripper.

The Robotiq's 3-finger joint positions, ranging from 0 (all the way open) to 255 (all the way closed), allow for IRLA to learn estimate joint positions on

Testing Results	Cylindrical	Cuboid	Flat	Amorphous	Hemispherical
Object Detection Correlation	0.15	0.6	0.03	0.70	0.23

Table 4.2: Summary of Robotiq object detection correlation.

a wide range of objects. The average joint positions for each finger differ based on the shape of the object as well as the final grasp orientation. The IRLA algorithm does not explicitly learn these features, and instead infers whether there is a correlation between these finger positions and the final grasp quality. The Robotiq object detection will not detect an object if the fingers are fully closed, revealing the relationship between the detection algorithm and the Robotiqs final finger positions. If an object is not detected, the fingers will be fully closed, but if an object is detected, the fingers final positions will lie in the previously mentioned range. The fingers have a 1-3mm error range when detecting the final finger positions. This source of error stems from the adaptive feature of the Robotiqs fingers, as the linkages cause the fingers to have a small amount of free movement at each movement step. This free-movement error is noted in Robotiq’s official documentation [39] and confirmed during lab testing.

Another hardware feature leveraged in IRLA is the Robotiqs adjustable closing force. This force also has a setting from 0, minimum force, to 255, the Robotiqs maximum applicable force [39]. The variance of the total range of force settings was deemed too large to effectively learn in a working lab setting. A brief survey of applications and force settings was undertaken to determine the key force points used in common applications at the NRG lab. The force

range was reduced to a binary setting, with -1 representing half force (force setting 122) and 1 representing full force (force setting 255). The grasping force was measured and determined to be within a $\pm 5\%$ from the Robotiq documentation through ten tests at the maximum force and ten tests at the half force settings. In training (Table 4.3), the two force settings performed almost the same, with only minor variations which can be attributed to the different training conditions between tests, grasp configurations, and limited testing examples.

Testing Results	Cylindrical	Cuboid	Flat	Amorphous	Hemispherical
Force Setting	0.44	0.47	0.52	0.44	0.49

Table 4.3: Summary of force setting correlation.

4.2 Software Framework

The software environment used for data collection and IRLA training includes Ubuntu and ROS. The code used for training the IRLA algorithm is based on C++11 standards. ROS was chosen for the ease of integrating multiple systems and collecting data easily. Various packages and sub-packages were employed or modified in this undertaking, all of which can be viewed in more detail in the Appendix. The software framework is developed off of the description in Section 3.2.

4.2.1 Robot Operating System (ROS)

ROS in this undertaking is only used for ease of implementation. The ROS community mainly uses either C++ or Python interchangeably. ROS communication is accomplished through a network consisting of ROSCORE, ROS nodes, and ROS messages. A package consists of one up to many interconnecting nodes that communicate through ROSCORE. ROSCORE is an underlying communication node that keeps track of the nodes, names, and communication lanes. ROS nodes send and receive messages through advertising/publishing messages on topics and subscribing to various other topics that are published from other ROS nodes. This aspect of ROS increases the potential to build modular code instead of large blocks of self-contained programs. As described in Section 3.2, four nodes are primarily used to gather data, control the Robotiq, and receive user feedback. Two nodes were explicitly created for implementing the IRLA algorithm, and the other two nodes are publicly available ROS programs. Both the Robotiq driver package and the Takktile driver package are publically available ROS packages.

The two new packages created for this effort are `sensor_integrator` (`Mechanical Control`) and `grasp_score` (`Scoring Control`). `Mechanical Control` is programmed as a ROS service that operates as a request/reply system. `Mechanical Control` takes in a request, shown below in Code 4.1, from `Scoring Control` to know when an object has been correctly placed within the gripper. The response from `Scoring Control` contains both the force setting needed for the current test and that the object is within the

```

bool ready
int32 force_setting
——
bool obj_detected
int32 finger_A_pos
int32 finger_B_pos
int32 finger_C_pos
int32 [] takk_sensors

```

Code 4.1: Request message passed from **Sorting Control** to **Mechanical Control**. **Sorting Control** sends the *ready* and *force_setting*. After **Mechanical Control** commands the **Robotiq** and records the sensor values, it responds with all of the information below the solid line.

gripper. When **Mechanical Control** (MC) is first run, MC sends a calibration request to the **Takktil** driver package before any object is placed within the gripper. Once MC has received a request from **Scoring Control** (SC), MC then requests the **Robotiq** driver package to close the gripper with the desired force and reads the response from the driver package on the following features: **Robotiq** joint positions, **Robotiq** object detection, and the force setting to confirm the grasp was executed successfully. MC then sends a message to the **Takktil** driver package to read the **Takktil** pressure sensor readings. MC then replies to SC with the request message, found in the lower portion of Code 4.1, which contains the **Robotiq** object detection, the **Robotiq** finger positions, and 36 **Takktil** pressure sensor readings.

Scoring Control, after receiving data back from **Mechanical Control**, then instructs the user to score the grasp by bumping, nudging, pulling, and

twisting with a reasonable amount of force. This portion of the testing is entirely dependent on the user’s environment and can be modified to better reflect the forces the object may encounter normally. The environment used for testing is a hazardous environment inside of a glovebox where robot manipulator speeds are limited and force control is used to maintain safety from crashes. The amount of force the gripper will encounter, if the object bumps into another object or the sides of the glovebox, are small due to the safety measures in use with robotic manipulators. Once the user has determined the quality of the grasp, the user enters his/her determination of grasp quality into SC and the feature weights of the scoring function are automatically updated (Equation 3.1). The quality of the grasp is defined as a score of -1 for complete loss of object, 0 for object movement within the gripper, and 1 for no visible movement. This process can be seen in Figure 4.4. All features and user responses are saved for future training and learning situations to include the ability in the future to combine these results with other testing metrics.

4.3 Test Objects and Results

Grasp Data Set per Object	
Training	50
Testing	5
Total	55
Total per Shape Type	165
Total Testing per Shape Type	15

Table 4.4: Each test object has a total of 55 grasp data sets. 5 data sets per object were reserved for testing IRLA after initial training.

```
Scoring Control is now Running.
Mechanical Control is Running.
Robotiq is broadcasting.
Takktille is broadcasting.
Waiting for user, all hardware is ready. Press Enter to continue.

Please enter the shape:
Cuboid
Please enter the desired force setting:
Full
Press Enter to continue.

Calibrating Takktille sensors.
Closing Robotiq at Full force.
Please enter the score:
1
Recording Sensor Values and updating weights.
Enter Y to continue training, N to stop.
```

Figure 4.4: Sample output from Scoring Control.

Objects were selected from the Nuclear and Applied Robotics Group (NRG) laboratory which are representative of objects encountered at Los Alamos National Laboratory (displayed in Figure 4.5 and listed in order in Table 4.5). The testing objects were chosen based on their similarities to objects found in glovebox environments. Various types of tools were also tested, but due to many tools needing a particular way of grasping to be of use or needing modifications to the tool itself to be correctly handled by the Robotiq, these were discarded. Interesting results were collated for the various different shapes and force settings below including a combined learning result garnered from combining the training results from each object. Each object was trained in 25 different positions inside of the Robotiq gripper at two different force settings. Each grasp configuration distribution per shape tested depends heavily on the object being gripped. Cylinders have fewer novel grasp configurations due to symmetry than hemispheres, while the number of novel

grasps for amorphous objects is nearly limitless. Each grasp was made to be as distinctly different from each other grasp already executed on the object as possible. Five additional data points per object were collected to serve as a testing sample pool (See Table 4.4). Only 165 out of the 275 grasp point data for cylindrical objects were used to ensure parity between learning per object type (randomly selected).

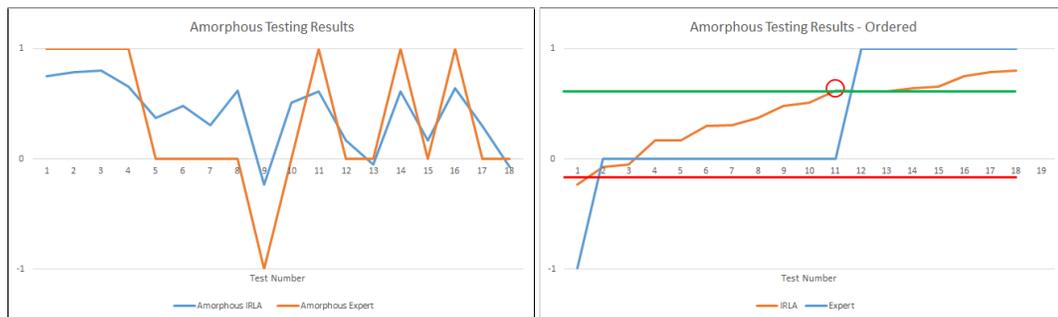


Figure 4.5: Object set used for testing and training purposes, arranged by shape. Shapes were chosen based on relationships between the shapes and glovebox tasks and to vary the sizes and properties.

Object List by Shape	
Cylindrical	Large Bolt, Aluminum Stock, Medium Radiation Canister Small Radiation Canister, Small Aluminum Part
Cuboid	Hard Rubber Brick, Eraser, Small Box of Bolts
Flat	Aluminum Block, CD with Sleeve, Small Wooden Piece
Amorphous	Foam Block, Rope, Bag of Foam Beads
Hemispherical	Large, Medium, and Small Bowls

Table 4.5: Listed object test set by shape in order of Figure 4.5

The range of response from the IRLA algorithm is continuous over the range of -1 to +1. While this range can be helpful for in-depth analysis of the scoring function, for the end-user it may not be helpful as it does not convey any extra information that is useful. For instance, if IRLA scores a grasp as 0.65, to what is that score in reference? What if the best grasp available for the object is 0.7? Fuzzy logic was applied to the response from IRLA to grade the responses as either *safe*, *uncertain*, or *unsafe* (See Figure 4.6). A curve was applied to the response range per shape type to differentiate the three responses IRLA can generate. The training data was run through the trained IRLA algorithm, and the curve was chosen to minimize false positives from the results and err on the side of caution for unsafe grasps.



(a) Amorphous test results unordered (15 data points)

(b) Amorphous test results ordered by IRLA score with false positive (green) and false negative (red) bars (15 data points). A false positive is circled in red.

Figure 4.6: Amorphous test results with visualized fuzzy logic separation of correctly labeled grasps.

False positives are defined when increasing the threshold for classifying

a result as a good grasp would remove more than two other correctly scored grasps. For instance, in the case in Figure 4.6a, IRLA's score is greater than the expert's score by more than 0.62. Three scores that were classified as a success by the algorithm were within the range of (0.58-0.61). Applying fuzzy logic in this case and moving the scoring threshold to 0.63 for validated grasps would eliminate three correctly identified grasp while also removing the false positive score of 0.62 from the one grasp. False negatives are defined similarly for Figure 4.6a, when IRLA's score is less than the expert's score by more than the absolute value 0.2 (the threshold between the scores when the expert scored grasps as -1 instead of 0). This logic is applied for each shape type to garner the results in Table 4.7.

For hazardous environments inside of a nuclear glovebox, a false positive is considered worse than a false negative, as the repercussions are potentially far worse if an object is dropped versus the need to re-grasp. Fuzzy logic is implemented after the training and testing results are finished. Using this definition and implementation of fuzzy logic, the IRLA algorithm training produced few false positives after expert selection of cutoff values.

Table 4.6 depicts The combined results (testing without object type knowledge), training the IRLA algorithm with all of the training samples and testing with 15 randomized data points from the testing pool results. The number of false positives and false negatives are greater than when IRLA is trained individually on each type of shape; however, this result does correctly score 53.333% of the results, showing the potential for generalizing IRLA to

work without shape information.

Testing Results	Combined
Average Grasp Quality	0.42
Average Generated Score	0.50
False Positives	5
False Negatives	6

Table 4.6: Combined results from all saved data, tested with 25 random samples (5 from each shape type). The shape parameter and weighting is ignored when inputting into IRLA.

Table 4.7 contains summarized data from the final testing for each type of shape. The average grasp quality is the average of the user’s scores for the shape, while the average IRLA score is the average IRLA scores generated when tested over the complete data set again. The closer these two numbers, the more IRLA has learned a scoring profile for each shape type. The average IRLA score also contains the probability of generating a false positive or false negative (if the average IRLA score is higher than the actual average score, there is a positive bias in the system which makes false negatives more common and *vice versa*).

Testing Results	Cylindrical	Cuboid	Flat	Amorphous	Hemispherical
Average Grasp Quality	0.73	0.80	0.13	0.40	0.87
Average IRLA Score	0.63	0.91	-0.08	0.41	0.77
False Positives (15)	1	0	0	1	0
False Negatives (15)	2	1	4	0	1

Table 4.7: Summarized training and testing results from objects in Figure 4.5

Five sets of testing data for each object were saved to validate the training results of IRLA. 75 total grasps were used as a testing set for IRLA.

The results in Table 4.7 from testing reveal several key learning metrics IRLA met. The number of false positives is either 0 or 1, while the number of false negatives was also minimized to 2 with exception of flat objects. The two false positives occurred from a testing sample for a cylindrical object and an amorphous object. Both of these samples were scored as unsafe (0) when IRLA scored them as good (passing the fuzzy logic cut-off for separation between good/unsure/bad). For the nuclear environment, minimization of false positives is the focus of training IRLA. Out of 75 testing grasps only 2 failed by this metric. The number of false negatives for all of the shapes is also low, 8 out of the 75 grasps, and combined with other information in Table 4.7 some conclusions can be drawn. In both the testing and training data there is a positive training bias for successful grasps (more training examples of successful grasps than failing grasps). This bias skews IRLA to differentiate good grasps from other grasps but does not learn as well to differentiate bad grasps from potentially unsafe grasps. A more evenly distributed training data set might help improve this statistic.

The cylindrical, cuboid, amorphous, and hemispherical objects tended to have similar results to each other in Table 4.7 for average scores from the testing data. The average scores for these four object types are also highly positive - reflecting the overall positive bias the training data created. However flat objects show a markedly reduced average score while also having the largest difference between IRLA and expert average scoring for the test data. Flat objects are both difficult to detect and difficult to avoid biased testing

compared to other shapes. Flat objects, when grasped firmly, fell into two categories of grasps - either pinched firmly between the fingers or encompassed in a basic grasp completely. Most other types of grasp for flat objects were able to be shifted easily when expert tested. It was noted that when picking planes/directions to attempt to move the object within the grasp, for flat objects it is hard to avoid picking a parallel force vector along the surface of the object. Flat objects also have the most false negatives out of the different object types, which can be attributed partly due to the above concerns.

The number of false negatives reveals further training and refining of the cutoff values is needed for failing grasps. The nature of linear combinations of features can cause over-training to occur. This can cause IRLA to focus too heavily on a few important features and lose the ability to score novel grasps correctly. Overfitting occurs when the scoring function models random error or noise instead of the relationship between features. The testing and training method used for IRLA is error prone and noisy due to its subjective nature. Keeping the training samples purposefully underfitted slightly avoids the data issue from greatly affecting the results while allowing room for online learning in the future.

4.4 Summary of Demonstrations and Analysis

This chapter contains the application of the inverse reinforcement learning algorithm as well as the training and testing method for the algorithm. The results are then analyzed for efficacy and room for future improvements.

The IRLA results from testing correctly identified grasps for 65 out of 75 trials. With the focus to train and differentiate between good grasps and other grasps (unsafe/failing), only 2 out of the 75 testing grasps were classified as a false positive.

IRLA's success rate demonstrate the possibilities of implementing machine learning in more areas than have been previously handled by trained technicians and hard-coded programs. Robots represent a reliable, and efficient future of automation, but programming robots and their grippers to recognize the same things their human counterparts do in their work area has long been sought after, and only recently been attainable in an easy to train method. The IRLA algorithm used in this paper is one step towards improving worker and site safety, especially in nuclear related areas in which the need for automation is mandated to keep radiation doses as low as reasonable achievable for operators.

Chapter 5

Conclusions and Future Work

General grasp validation - the ability to determine the quality of a grasp and its likelihood to fail - is a critical step in handling hazardous materials with robotic manipulators. A failure can destroy a one-of-a-kind prototype or incur a high cost to clean the affected area if the material is radioactive or chemically active. IRLA - Inverse Reinforcement Learning Algorithm - was implemented to quantify grasp quality on common objects inside of a glovebox. The results show that training an inverse reinforcement learning algorithm was useful in determining the quality of a grasp on the selected objects. IRLA identified 97.33% of tested grasps correctly between safe and unsafe grasps. After training and testing the set of objects, an attempt was made to examine the effect of removing the primitive shape of an object from the identifier for an object. Removing the identifier led to a 53.333% successful quantification of grasp quality on the testing samples, showing the potential to learn a general set of characteristics for a safe grasp.

5.1 Summary of Covered Topics

The **Introduction** describes grasp quantification and the context in which it is learned. Grasp validation or grasp quality is translating a grasp score into a form a normal operator can understand - whether a grasp is designated as *good*, *bad*, or *potentially unsafe*. Determining the grasp quality of all objects is intractable, but limiting the objects to common objects in a task space resolves this issue, and this approach is feasible in the glovebox manufacturing domain given the control over what items will ever be found in a given glovebox. A machine learning algorithm is implemented to quantify grasps for objects within a nuclear glovebox environment.

A thorough review of previous and current efforts in the fields of grasp verification and grasp planning was undertaken in the **Literature Review** chapter. Early work consists of grasp planners using human-input to define object locations which were then integrated with primitive tactile sensors to detect and grasp an object. Advancements of technology facilitated advancements on approaches to grasp planning and verification. 3D vision systems and faster processing speeds allowed for simulating of physical systems and objects. Current grasp planners attempt to leverage these two capabilities with increasing success. While an increase in research activity on the topic of grasp planning has been observed, grasp validation has remained relatively simplistic or addressed heuristically. Many algorithms pertaining to grasp quality first generate known good grasps on objects based on models of existing objects and the manipulating gripper. While these efforts are robust methods to

generate effective grasps, these algorithms currently have few ways of successfully scoring grasps after grasp execution without a resource-intensive physics simulation. This thesis attempts to leverage machine learning techniques to train a system to recognize, via various sensors, when a *good* grasp has been executed and minimize the number of *potentially unsafe* grasps.

The Inverse Reinforcement Learning Algorithm is then detailed in the **Learning Solution and Data Gathering** chapter. Iterative improvement is first described, followed by various modifications to a learning algorithm approach. Features are described and their relations to the learning problem detailed. Modifications to a preference perceptron and interaction model are made to apply an inverse reinforcement learning algorithm (IRLA) to the task space in this thesis. The two developed software packages, **Mechanical Control** and **Sorting Control**, decrease the training time necessary and automate the training process. IRLA integrates user-determined features and user feedback to determine the grasp quality of executed grasps.

In **Demonstration and Analysis** the final hardware selection, training procedures, evaluation methodologies, and experimental results are presented. The hardware incorporates the Robotiq 3-Finger Adaptive Gripper and Takktile pressure sensors to gather feature data. The recorded feature data and user feedback trains the software-implemented IRLA algorithm. Test data is recorded and applied to the IRLA algorithm after training. The IRLA results show 2 out of 75 testing grasps were classified as a false positives. False positives - an unsafe grasp quantified as a safe grasp - are identified as

the main testing metric for grasp validation. False positives have the highest probability of causing damage to the object. Identifying, and learning from false positives and critical failures is the goal for any future development efforts on the IRLA’s use for grasp validation.

5.2 Application

IRLA is not a complete grasping architecture. Instead, IRLA fills a critical safety role between grasp planning and the ensuing manipulation and trajectory tasks. IRLA recognizes potentially unsafe grasp executions and trigger an object release and re-plan method. Learning algorithms can be taught to recognize when a planned grasp is unstable, dangerous, differs from planned grasps, or is categorically safe. By recognizing these grasping states, learning algorithms decrease compounding error as tasks are completed in a manipulation planning tree. Reducing the uncertainty in planning algorithms allows for more intricate planning steps while ensuring safe operation.

5.3 Future Work

IRLA can be extended and incorporated as an independent working part of most grasp planning solutions. The approach covered in this thesis includes training and testing on a specific object set with simple geometries, but this approach can be extended to include more complex shapes with primitive shape decomposition of objects. This approach would separate the object into primitive shapes, identify a specific primitive shape for a grasp planner

to utilize, and then quantify the grasp quality for the selected grasp. Further verification work should be done on a common standard set of testing objects such as the YCB Object and Model Set [7].

Another possible extension of IRLA would be to test the algorithm using different sensor suites and different gripper models. This testing would further validate the hardware-agnostic approach presented above. Different sensor suite extensions can be utilized including vision-systems, infrared depth sensors, force-torque sensors, and a denser set of tactile pressure sensors. Typical tactile sensing pads on humanoid hands contain hundreds of tactile sensing regions, increasing the resolution of tactile sensors allows for other developed grasp validation methods to be combined with IRLA. The geometry of activated sensors can be leveraged to model the position of the object in the gripper. Vision-systems can then verify this model, or act independently to recognize the quality of the executed grasp.

One limitation of this implementation of inverse reinforcement learning is the linear combination of features. Increasing the complexity of the feature weights would allow for more inference to be made on each individual feature, although increasing the complexity increases the possibility of over-training the system by including noise and error. This possibility can be reduced by implementing a rolling, online system in which older training results are removed from the weights or by decreasing the weight updating effect via implementing a decreasing learning rate variable. A further extension of including history or a learning rate would be to prune examples that did not result in a weight

change or if the feature readings are the same as another example.

Grasp validation can be solved by forming a grasp database for every confirmed valid grasp. A database is a more training-intensive method of quantifying good versus bad grasps. This approach cannot handle novel grasps well, as the underlying principle relies only on empirical data. A small difference between two grasps might have completely different physics involved depending on object shape or other factors. Accounting for the large range of responses from various sensors in addition to approximating between data points is an endeavor that may be viable for certain object sets/tasks. Integrating IRLA with a database, either to generate the database or learn from a database of grasps, is a future possibility.

To make IRLA an online algorithm (an algorithm that is able to learn from new data while being used in tasks) IRLA needs to recognize automatically when a grasp has failed or if a grasp is either categorized as a false positive or false negative. When a grasp has failed and is detected, the IRLA algorithm could use the previously recorded shape and feature information to update its scoring algorithm and continue learning as tasks continue. As data accumulates, examples that were outliers will be reduced in importance and may be solved completely with more training examples.

Currently the IRLA algorithm as trained and employed was developed for a specific number of features and expert feedback, but IRLA can be generalized in its ROS implementation to take in data and user feedback to generate the scoring algorithm. With this adjustment, users can use their own sensor

suite and gripper combinations to develop custom learned libraries for object sets pertaining to their field of work. Automatically applying fuzzy logic should be a more automated process that can be as effective as a human or better. A set of questions can determine the aim of the user and apply the correct optimization.

Human-Robot interaction is a new and broadening field of research attempting to teach robots how to work with or around human counterparts and vice versa. The hand-off - or transfer of an object from a human to a robot or vice versa - will be needed for numerous envisioned cooperative tasks. Detecting a safe state where a human has a firm hold of an object and where the robot may then release the object without dropping it may be difficult to discern using normal methods. IRLA has the capability of learning a hand-off scoring function that can further differentiate the quality of the grasp for transfer based on variety of parameters including: the characteristics of the user, the robot, or the object itself. Such a score could even account for its relative value, and, for example, allow for the improved efficiency of rapidly transferring cheap or hard-to-break items and yet ensure items like a glass of water are more conservatively transferred even with the expense of speed.

5.4 Concluding Comments

In this thesis, an inverse reinforcement learning algorithm was trained to quantify the quality of robotic grasps on selected objects. The machine learning algorithm employed for this task is currently only trained for use

with specific items for deployment in a nuclear environment, but the results show that this approach can be utilized today to increase safety when handling any type of material or object. Once a set of sensors have been decided upon, integration into a laboratory setting and training should take under a week to utilize IRLA to increase safety in a task. Machine learning typically only exists in software or research laboratories, but after review machine learning can be implemented today in industry to both increase worker safety and reduce grasping error when manipulating objects.

Bibliography

- [1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship Learning via Inverse Reinforcement Learning. In *International Conference on Machine Learning*, 2004.
- [2] Adam Allevato. An object recognition and pose estimation library for intelligent industrial automation. Master's thesis, The University of Texas at Austin, Austin, TX, 5 2016. NRG Lab.
- [3] Antonio Bicchi, J Kenneth Salisbury, and Paolo Dario. Augmentation of grasp robustness using intrinsic tactile sensing. In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pages 302–307. IEEE, 1989.
- [4] Cheryl Brabec. A shape primitive-based grasping strategy using visual object recognition in confined, hazardous environments. Master's thesis, The University of Texas at Austin, Austin, TX, 12 2013. NRG Lab.
- [5] Randy C. Brost. Planning robot grasping motions in the presence of uncertainty. Technical Report CMU-RI-TR-85-12, Robotics Institute, Pittsburgh, PA, July 1985.
- [6] Randy C Brost. Automatic grasp planning in the presence of uncertainty. *The International Journal of Robotics Research*, 7(1):3–17, 1988.

- [7] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols. *arXiv preprint arXiv:1502.03143*, 2015.
- [8] Giovanni Claudio, Fabien Spindler, and François Chaumette. Vision-based manipulation with the humanoid robot romeo. In *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*, pages 286–293. IEEE, 2016.
- [9] KELLY J Cole and JAMES H Abbs. Coordination of three-joint digit movements for rapid finger-thumb grasp. *Journal of neurophysiology*, 55(6):1407–1423, 1986.
- [10] Mark R Cutkosky and Robert D Howe. Human grasp choice and robotic grasp analysis. In *Dextrous robot hands*, pages 5–31. Springer, 1990.
- [11] Hao Dang, Jonathan Weisz, and Peter K Allen. Blind grasping: Stable robotic grasping using tactile feedback and hand kinematics. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5917–5922. IEEE, 2011.
- [12] Colin Davidson and Andrew Blake. Caging planar objects with a three-finger one-parameter gripper. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 3, pages 2722–2727. IEEE, 1998.

- [13] Mathew DeDonato, Velin Dimitrov, Ruixiang Du, Ryan Giovacchini, Kevin Knoedler, Xianchao Long, Felipe Polido, Michael A. Gennert, Takn Padr, Siyuan Feng, Hirotaka Moriguchi, Eric Whitman, X. Xinjilefu, and Christopher G. Atkeson. Human-in-the-loop control of a humanoid robot for disaster response: A report from the darpa robotics challenge trials. *Journal of Field Robotics*, 32(2):275–292, 2015.
- [14] Rosen Diankov, Siddhartha S Srinivasa, Dave Ferguson, and James Kuffner. Manipulation planning with caging grasps. In *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, pages 285–292. IEEE, 2008.
- [15] Mehmet Dogar and Siddhartha Srinivasa. A framework for push-grasping in clutter. *Robotics: Science and systems VII*, 1, 2011.
- [16] Strahinja Dosen, Marko Markovic, Matija Strbac, Minja Perovic, Vladimir Kojic, Goran Bijelic, Thierry Keller, and Dario Farina. Multichannel electrotactile feedback with spatial and mixed coding for closed-loop control of grasping force in hand prostheses. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2016.
- [17] David Fischinger and Markus Vincze. Empty the basket-a shape based learning approach for grasping piles of unknown objects. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2051–2057. IEEE, 2012.

- [18] Columbia University Robotics Group. Graspit!, 2002. Accessed: 2016-05-20.
- [19] Paul Hebert, Nicolas Hudson, Jeremy Ma, and Joel Burdick. Fusion of stereo vision, force-torque, and joint sensors for estimation of in-hand object location. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5935–5941. IEEE, 2011.
- [20] Shinichi Hirai et al. Design and analysis of a soft-fingered hand with contact feedback. *IEEE Robotics and Automation Letters*, 2016.
- [21] Thea Iberall. Grasp planning from human prehension. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI’87*, pages 1153–1156, San Francisco, CA, USA, 1987. Morgan Kaufmann Publishers Inc.
- [22] Ashesh Jain, Brian Wojcik, Thorsten Joachims, and Ashutosh Saxena. Learning Trajectory Preferences for Manipulators via Iterative Improvement. *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, 2013.
- [23] Jyh-Shing Roger Jang and Chuen-Tsai Sun. *Neuro-fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.
- [24] Brendon John, Jackson Carter, Javier Ruiz, Sai Krishna Allani, Saurabh Dixit, Cindy M Grimm, and Ravi Balasubramanian. Human-planned

robotic grasp ranges: Capture and validation. *arXiv preprint arXiv:1607.03366*, 2016.

- [25] Uikyum Kim, Dong-Yeop Seok, Yong Bum Kim, Dong-Hyuk Lee, and Hyouk Ryeol Choi. Development of a grasping force-feedback user interface for surgical robot system. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 845–850. IEEE, 2016.
- [26] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement Learning in Robotics: A Survey. In *International Conference on Machine Learning*, 2013.
- [27] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154. IEEE, 2004.
- [28] Cecilia Laschi, Philippe Gorce, Juan-Lopez Coronado, Fabio Leoni, Giancarlo Teti, Nasser Rezzoug, Antonio Guerrero-González, Juan Luis Pedreño Molina, Loredana Zollo, Eugenio Guglielmelli, Paolo Dario, and Yves Burnod. An Anthropomorphic Robotic Platform for Experimental Validation of Biologically-inspired Sensory-motor Co-ordination in Grasping. In *International Conference on Intelligent Robots and Systems*, 2013.
- [29] Qujiang Lei and Martijn Wisse. Object grasping by combining caging and force closure. In *Control, Automation, Robotics and Vision (ICARCV), 2016 14th International Conference on*, pages 1–8. IEEE, 2016.

- [30] Efrain Lopez-Damian, Daniel Sidobre, Stephane DeLaTour, and Rachid Alami. Grasp planning for interactive object manipulation. In *Proc. of the Intl. Symp. on Robotics and Automation*, 2006.
- [31] Microsoft. Microsoft xbox one kinect sensor. http://www.newegg.com/Product/Product.aspx?Item=N82E16874103418&cm_re=kinect_-_74-103-418_-_Product. Accessed: 2016-05-20.
- [32] Andrew T Miller, Steffen Knoop, Henrik I Christensen, and Peter K Allen. Automatic grasp planning using shape primitives. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 2, pages 1824–1829. IEEE, 2003.
- [33] Newegg.com. Asus xtion pro live. <http://www.newegg.com/Product/Product.aspx?Item=N82E16826785030>. Accessed: 2016-05-20.
- [34] Richard Paul. Modelling, trajectory calculation and servoing of a computer controlled arm. Technical report, DTIC Document, 1972.
- [35] R. Pelosof, A. Miller, P. Allen, and T. Jebara. An svm learning approach to robotic grasping. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 4, pages 3512–3518 Vol.4, April 2004.
- [36] Nancy S Pollard. Synthesizing grasps from generalized prototypes. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 3, pages 2124–2130. IEEE, 1996.

- [37] Elon Rimon and Andrew Blake. Caging 2d bodies by 1-parameter two-fingered gripping systems. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 2, pages 1458–1464. IEEE, 1996.
- [38] Elon Rimon and Andrew Blake. Caging planar bodies by one-parameter two-fingered gripping systems. *The International Journal of Robotics Research*, 18(3):299–318, 1999.
- [39] Robotiq. 3-finger adaptive robot gripper instruction manual, 2016. [Online; accessed February 20, 2016].
- [40] Robotiq. Adaptive robot gripper 2-finger, 2016. [Online; accessed February 20, 2016].
- [41] Joseph M Romano, Kaijen Hsiao, Günter Niemeyer, Sachin Chitta, and Katherine J Kuchenbecker. Human-inspired robotic grasp control with tactile sensing. *IEEE Transactions on Robotics*, 27(6):1067–1079, 2011.
- [42] National Park Service. Nts - emad facility 009, 1997. [Online; accessed March 05, 2016].
- [43] Andrew Sharp, Matthew Horn, and Mitchell Pryor. Operator Training for Preferred Manipulator Trajectories in a Glovebox. In *IEEE Workshop on Advanced Robotics and its Social Impacts*, 2017.
- [44] Ioan A. Sucas and Sachin Chitta. MoveIt! "<http://moveit.ros.org/about/>". Accessed: 2015-07-21.

- [45] Ioan A. Sucas and Sachin Chitta. MoveIt! Robots. "<http://moveit.ros.org/robots/>". Accessed: 2015-07-21.
- [46] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.
- [47] Siyi Tang, Rohan Ghosh, Nitish V Thakor, and Sunil L Kukreja. Live demonstration: Real-time orientation estimation and grasping of household objects for upper limb prostheses with a dynamic vision sensor. In *Biomedical Circuits and Systems Conference (BioCAS), 2016 IEEE*, pages 135–135. IEEE, 2016.
- [48] Andreas ten Pas and Robert Platt. Using geometry to detect grasp poses in 3d point clouds. In *Intl Symp. on Robotics Research*, 2015.
- [49] Yaroslav Tenzer, Leif P. Jentoft, and Robert D. Howe. Inexpensive and Easily Customized Tactile Array Sensors using MEMS Barometers Chips. In *Nuclear Science Symposium and Medical Imaging Conference*. IEEE, 2012.
- [50] A. W. Thompson. Nuclear residues repacking glovebox, building 440, rocky flats nuclear weapons plant, 2002. [Online; accessed March 03, 2016].
- [51] A. W. Thompson. Meka robotics' humanoid torso and anthropomorphic hands, 2009. [Online; accessed March 06, 2016].

- [52] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.
- [53] A Frank van der Stappen, Chantal Wentink, and Mark H Overmars. Computing immobilizing grasps of polygonal parts. *The International Journal of Robotics Research*, 19(5):467–479, 2000.
- [54] ZhiDong Wang and Vijay Kumar. Object closure and manipulation by multiple cooperating mobile robots. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 1, pages 394–399. IEEE, 2002.

Index

Abstract, vi
<i>Acknowledgments</i> , v
<i>Bibliography</i> , 83
<i>Conclusions and Future Work</i> , 67
<i>Dedication</i> , iv
<i>Demonstration and Analysis</i> , 48
<i>Introduction</i> , 1
<i>Learning Solution and Data Gathering</i> , 35
<i>Literature Review</i> , 15

Vita

Matthew W. Horn was born in Layton, Utah on October 8th 1991, the son of William T. Horn and Dana L. Horn. He received the Bachelor of Science degree in Mechanical Engineering from Rice University Cum Laude. He applied to the University of Texas at Austin for enrollment in their nuclear robotics program. He was accepted and started graduate studies in August, 2014.

Email address: mwhorn1008@gmail.com

This thesis was typeset with \LaTeX^\dagger by the author.

[†] \LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's \TeX Program.