The Thesis Committee for Cassidy Morgan Elliott
Certifies that this is the approved version of the following thesis:

Compliance-based Affordance Templates for Remote Mobile Manipulation

APPROVED BY

SUPERVISING COMMITTEE:

Mitch Pryor, Supervisor

Eric van Oort, Co-Supervisor

Compliance-based Affordance Templates for Remote Mobile Manipulation

by

Cassidy Morgan Elliott

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

December 2020

Dedication

To my friends and family, whose support means the world to me.

Acknowledgments

First and foremost, I would like to thank my parents and my loving boyfriend, without whose support I would not be where I am today. I would like to thank Dr. Mitch Pryor for his advice and technical guidance throughout my graduate school career. I would also like to thank Dr. Eric van Oort and the RAPID Consortium for their excellent technical feedback on this work. Finally, I would like to thank all of my collaborators at UT, Woodside, NASA, and TRACLabs for their help in this project, with a special shout out to Anthony Biviano and Phil Strawser for making Perth feel like home.

This work was financially supported by Woodside Energy, the University of Texas at Austin, and UT's generous fellowship donors. Additional thanks to NASA and TRACLabs for granting me access to their software packages.

Abstract

Compliance-based Affordance Templates for Remote Mobile Manipulation

Cassidy Morgan Elliott, M.S.E.
The University of Texas at Austin, 2020

Supervisor: Mitch Pryor
Co-Supervisor: Eric van Oort

This thesis details the implementation of Affordance Templates with a compliance controller to pseudo-autonomously perform complex contact tasks in industrial environments. Multiple action planning methods were evaluated, and ATs were chosen as the best option for industry use due to their intuitive user interface. Two Affordance Template packages were implemented and evaluated to determine the package best suited for use by novice operators. That package was then implemented in conjunction with a compliance controller in the form of a remote demonstration to showcase reduction in operator cognitive burden by automatically managing contact forces and significant command latency (approx 250 ms) during task performance. The results showed that ATs greatly reduced task execution time and number of errors compared to manual control, with added compliance reducing AT set up time. Finally, this thesis introduces Affordance Primitives, which were created to increase

the flexibility of Affordance Templates and present a path for future development in the field of automated task planning.

Table of Contents

List of Tables

List of Figures

Acronyms

ADL Action Description Language. 8, 16, 17, 25

AP Affordance Primitive. vi, 86, 88–90, 92, 93, 96–99, 104, 106, 108

AR augmented reality. 73

AT Affordance Template. vi, vii, xi, 7, 9–13, 16–23, 25–27, 32–39, 42, 49, 50, 58–63, 66, 67, 71, 73, 75, 78–91, 95–99, 101–108

CRAFTSMAN CaRtesian-based AFfordance Template Suite for MANipulation. 9, 11, 12, 19, 34, 36–40, 42, 43, 47–51, 59–61, 102, 108–110

DARPA Defense Advanced Research Projects Agency. 19

DBB double block and bleed. xi, xii, 4, 6, 61, 62, 64, 71–74, 77–84, 103

DRC DARPA Robotics Challenge. 19–22, 36, 47

EEF end effector. 6, 7, 9–11, 17, 23, 26–33, 45–47, 49, 57, 64, 66, 67, 69, 75, 76, 79–83, 85, 87, 90, 93, 97, 98, 100–102, 106

F/T force torque. 10, 31, 32, 64–66

GUI graphical user interface. xii, 17, 22, 32, 35, 37, 38, 48–53, 57, 60, 68, 70, 73, 82, 101

IK inverse kinematics. 47, 59

JSC Johnson Space Center. 2, 19, 37

KDL Orocos Kinematics and Dynamics Library. 47

LNG Liquid Natural Gas. 3, 4, 12, 49, 61, 62, 71

NASA National Aeronautics and Space Administration. v, 2, 9, 19, 20, 22, 36, 37, 109

NRG Nuclear and Applied Robotics Group. 2, 62, 67

OAC Object-Action Complex. 26

ROS Robot Operating System. 8, 18, 20, 23, 27, 36, 37, 42, 53, 57, 107, 109, 110

Rviz ROS Visualizer. 8, 37, 38, 40, 42, 46–51, 73–75, 82

Chapter 1

Introduction

As robot manipulation capabilities improve, industries are seeking to incorporate robots into their workforce. Robots have the potential to perform a wide variety of manipulation tasks currently completed by human employees. These tasks range from potentially dangerous, including handling high-voltage equipment, to tasks that are simply tedious for dedicated human operators, such as repeated actuation of a switch or valve. Such tasks are commonly referred to as the "3 D's": Dirty, Dangerous, or Dull [36].

Despite advancements in manipulation, robots still struggle to autonomously perform tasks that require precision or must manage forces that vary throughout the task. Examples of such contact tasks include turning a doorknob, flipping a switch, rotating a valve, etc. These tasks require some level of precision and for the robot to apply a force whose direction and magnitude changes during the task. If the robot is to complete the task autonomously, then it must also account for uncertainty in task location due to sensor noise or biases. Failure to complete the task correctly can damage the manipulated object, the robot, or both, which can be dangerous and expensive depending on the scenario.

While manipulation tasks can be completed via manual teleoperation, this

implementation is extremely tedious for the operator as all motions must be defined each time the task is performed. To appropriately manage the forces, the operator must count on experience or more complicated haptic feedback, which is not always available. Thus, the only employees currently capable of completing complex manipulation tasks via teleoperation are those with expertise in robotics, and these are not the employees utilizing the robots on the front lines.

The aim of this research is to develop and refine more intelligent tools to allow human operators to complete complex manipulation tasks more quickly and with less difficulty than with pure teleoperation. With improved situational awareness and more intuitive control options, we enable more users to successfully complete complex manipulation tasks remotely. Without the barrier to entry of dedicated robotics experts permanently stationed on site, it will be easier and more attractive for industries to use robots to perform identified "3 D tasks" using remote systems.

1.1 The Woodside Project

This thesis was motivated by a collaborative project involving the University of Texas at Austin's Nuclear and Applied Robotics Group (NRG), NASA's Johnson Space Center (NASA-JSC), and Woodside Energy (Australia's largest natural gas producer). The energy industry, in particular, is interested in increasing automation in their facilities. Oil and Natural Gas (O&NG) plants are potentially dangerous places to work with numerous stored superheated, pressurized liquids and/or gasses. Routine maintenance tasks must be performed by employees to prevent potential catastrophe. The combination of a hazardous workplace with repeatable, menial

tasks is the perfect environment to implement robotic or remote solutions that can complete the work autonomously or with a greater degree of autonomy than direct teleoperation.

Woodside's Intelligent and Autonomous Systems group's mission is to automate certain surveillance, maintenance, and emergency response tasks on their liquid natural gas (LNG) facilities in an effort to reduce the number of personnel that routinely face occupational hazards.

To deploy robots to remote assets in order to perform these tasks in place of human employees, Woodside must be confident that their robotic systems can successfully perform a variety of complex tasks while supervised and/or controlled remotely. Additionally, any robots that are deployed must not become a liability or require unscheduled human intervention, or else they are potentially creating more hazards around the plant than they are alleviating. Finally, the proposed solution must be more economically attractive than the current implementation; the robots need to be used routinely and make current workers more efficient for companies to want to invest in them as long-term solutions.

1.1.1 Common LNG Maintenance Tasks

In this context, a task refers to a series of actions required to complete a goal. Tasks can be categorized as surveillance, navigation, or manipulation/contact tasks. Currently, surveillance and navigation tasks are the most sophisticated, while manipulation tasks - particularly contact tasks - still frustrate experienced operators.

Common tasks that a robot at an LNG plant could complete include perform-

Figure 1.1: Woodside employees perform surveillance tasks on their Pluto facility using their robot Ripley.

ing pipe maintenance, monitoring plant emissions, testing for the presence of currents in wires, etc. These tasks primarily consist of contact tasks, which are the most complex, difficult tasks to perform remotely as there is a greater dependence on sensor feedback than with surveillance and navigation tasks.

An example of a common LNG contact task is a double isolation and bleed or "double block and bleed" (DBB) procedure. This procedure is performed to isolate specific downstream equipment so that maintenance can be performed without shutting down the plant or process. To perform the isolation, three valves must be turned in a specific order. First the two block valves are closed to isolate the downstream equipment, then the bleed or vent valve between the block valves is opened to vent the fluid to a safe location. To warrant the use of the double isolation procedure,

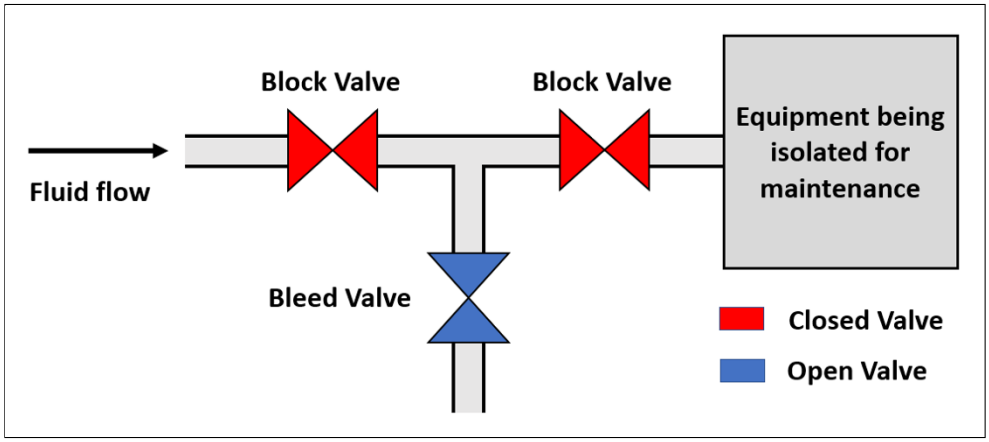


Figure 1.2: Diagram of a Double Block and Bleed Procedure

the fluid being vented is often pressurized, superheated, or toxic, making this task an ideal one to automate.

From a robotic standpoint, this is a difficult task for the user to perform. First, the user must ensure that the robot is close enough such that all three valves are reachable and visible to the onboard sensors. Additionally, the valves may not be co-planar or turn in the same direction making the task more difficult for the operator to conceptualize and potentially reducing the operator's situational awareness during task performance. The user must ensure the manipulator exerts enough force to turn the valves without exerting too much force, which would cause the low-level controllers to safety stop the robot. Finally, the hardware used in DBB is not frequently used. Thus its operational state is often not understood due to a lack of experience and data related to the specified valves.

When performing a manipulation task, the Operator's cognitive burden increases sharply with each additional action added to the task sequence. Thus, in-

dustries aim to introduce as much automation as possible when completing complex tasks such as a DBB.

1.2 User Control Modalities

When teleoperating a robot, there are two primary direct control methods that the user may be utilizing, point-to-point planning and jogging, both of which come with unique challenges. It is important to note that for both cases the EEF position and orientation are the only aspects of the manipulator that the user has any control over. The individual joint angles and overall configuration are determined by the inverse kinematics algorithm.

First, the operator may be planning via a point-to-point method. In this method, the operator picks a desired pose and orientation for the End-Effector (EEF) to reach, and a motion planner solves for a trajectory between the starting point and the desired end point. For point-to-point planning, known obstacles in the environment can be accounted for so that collisions are prevented, but unknown obstacles become even more difficult to handle because the user has limited control over the generated trajectory. Depending on the solver, that trajectory might or might not be the most efficient, and could lead the arm to twist in an unexpected way, potentially colliding with temporary obstacles in the environment. Dealing with temporary obstacles requires the operator to either change their starting position, add the obstacle to their list of known permanent obstacles, or plan in increments, all of which are inconvenient.

The second user control option is jogging, or real-time control. In this method,

the user commands the position or velocity of the manipulator continuously using an input device such as a joystick, 3D mouse, or motion tracker. Thus, the user has complete control over the path the EEF takes, but this also means that the user's cognitive burden for performing the task is much greater as they must continuously control the manipulator. While the operator has greater control over the trajectory, jogging does not allow for the same collision-avoidance safety measures as point-to-point planning. Additionally, if the operator is jogging the manipulator, they have to guide the arm along the desired trajectory (which could be anything from a straight line to an arc or circle) while also ensuring the arm does not twist itself into a poor configuration (one that puts the arm in collision with itself or any obstacles in the environment), on top of managing the forces at the point of contact.

As mentioned above, contact tasks place a greater burden on operators than the other two task types due to the need to monitor forces and ensure safe operation [9]. To make robots more attractive to potential industry partners, the goal is to shift from direct user control to supervisory control, which reduces operator burden significantly. By getting rid of the repetitive aspect of task setup/performance, actions and tasks can be performed repeatedly with ease.

1.3 Affordance Templates

Affordance Templates ATs are constructs that allow for the quick implementation of multi-step manipulation tasks. Coined by TRACLabs as a "task representation and execution framework" [37], Affordance Templates consist of a 3D model of an object and a predefined action sequence performed after an operator aligns the

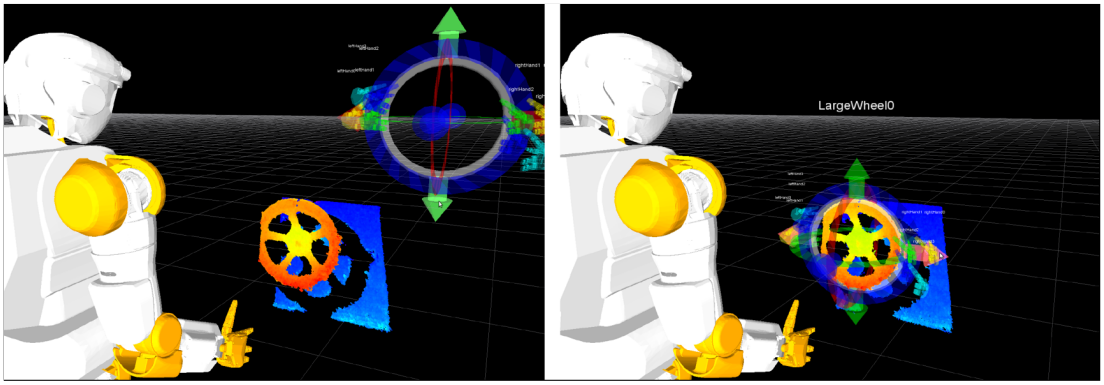


Figure 1.3: Valkyrie using an Affordance Template to turn a wheel [13]

model with the object's sensor-based 3D point cloud [14, 33]. Once aligned, the task can be autonomously completed by executing a sequence of sub tasks, defined using Action Description Language (ADL), which is described in Chapter 2, or some other construct. The template is defined once for each task for a given object, and used *ad infinitum* when the task needs to be performed again. This ensures task repeatability, minimizes the impact of sensor uncertainty, and reduces the cognitive burden on the user.

At run time, the 3D model is loaded into the Robot Operating System (ROS) visualizer (Rviz) containing the depth cloud data for onboard sensors. The operator (or possibly another autonomous agent) then aligns the model with its real-world counterpart with respect to the manipulator [13], as shown in Figure 1.3. Thus, the only adjustments that need to be made to the template after its initial creation are the size, location, and orientation of the model in Rviz [18]. This reduces the setup time for the task from minutes to seconds and allows novice users to quickly and routinely perform complex contact manipulation tasks.

ATs have several limitations. First, they rely on potentially noisy sensor data to align the virtual model. Due to tight AT tolerances, even a slight misalignment is likely to cause the EEF to exceed desired force/torque limits, causing the task to fail. Another downside to using ATs is that they cannot be paused or resumed; if a task fails at any point during the template's execution, the robot and the object must be completely reset before being reattempted. Finally, Affordance Templates are computationally expensive and limited in their scope. This means that only one manipulation object can be loaded into the virtual environment at a time, which limits the current usefulness of Affordance Templates.

Another issue is that there are only two known implementations of ATs currently available: TRACLab's CRAFTSMAN and NASA's newer UseIt. While a previous version of CRAFTSMAN is publicly available, neither implementation is free and open source. Both packages are being actively developed. Thus, their evaluation by the community is limited. These and related efforts are discussed in more detail in Chapters 2 and 3.

Yet another issue is one of scope, which refers to the duration and complexity of tasks captured in a single template. For example, should an AT manage just a door handle; or encompass the approach to the door, checking if it is locked, turning the knob, opening the door, moving through the door, and then closing the door. The answer likely lies in between these two extremes and the approach to scope will impact the design of any AT generating/execution software packages.

Finally, ATs may prove difficult to scale if the number of tasks is large. They were originally developed for NASA where the set of tasks in an environment such as

the space station are finite and known *a priori*, but this is not true if they are used in more complex and ever-changing environments. Thus there is a need to either rapidly develop ATs or provide the user with a way to define a quick equivalent for one-off tasks or to record the completion of a task and - from that - define and save the resulting template.

1.4 Compliant Control

To increase the flexibility of Affordance Templates and reduce the number of failures due to misalignment, UT Austin has integrated ATs with a compliance controller with the goal of increasing the positional tolerance of ATs when performing a task while also automatically managing forces being applied by the EEF to the object. The compliant controller utilizes an impedance model, which controls the relationship between the force and velocity of the EEF [17]. Thus, the compliance controller uses the feedback from the force torque (F/T) sensor mounted on the manipulator to regulate the velocity of the arm to keep the force within desired limits [42]. Impedance control is commonly used to make rigid manipulators "human-friendly" or combined with series-elastic actuators for more strategic interactions. When a force is sensed, the impedance can be set so that the arm moves to minimize that force as prescribed by a set of spring and damping virtual parameters. Thus, if a person is in the way of the arm, the arm will simply slow to a stop, but if the person is pulling or pushing the arm, the velocity changes to move in the direction of the force. The same principle applies for performing contact tasks, but the force threshold is not zero. By setting this threshold higher than the force necessary to

perform the task but lower than the arm's force limit, the controller manages the forces on the manipulator, removing that burden from the operator.

The compliance controller used for this project is unique in that it allows for the compliance parameters to be set separately for each degree of freedom and for the compliance parameters to be changed at any time. This means that the manipulator can be stiff in one dimension while being completely compliant in another, and this can be changed during task execution to fit the needs of each step of the task. For example, when turning a valve, the same compliance parameters may or may not be optimal when initially grasping the valve as when turning the valve.

Without compliance, if the arm was moving such that it collided with a rigid object in the environment, the arm would exert a force on that object that might be large enough to pose a danger to the arm, object, or both. With compliance enabled, the arm is prevented from exerting a force larger than is safe. Additionally, the alignment of a template to a task object without compliance must be extremely precise, which might not be possible depending on the accuracy of the sensor. Compliance allows for affordance templates to be more robust to uncertainty and eliminates the need for the user to manage the EEF forces manually while performing the task.

1.5 Summary of Objectives

With this thesis, I aim to show that the implementation of Affordance Templates combined with a compliant Cartesian jogger reduces operator burden while performing contact tasks. To do this, I will implement two existing AT software packages, UseIt and CRAFTSMAN, and compare and contrast their features. I will

also implement the compliant Cartesian jogger with these Affordance Templates and demonstrate the increased flexibility of ATs from this combined AT-compliant jogger approach. While task-agnostic, the system(s) are evaluated completing a double-block-and-bleed (DBB) contact task, which is representative of a variety of tasks found in the nuclear, drilling, refining, LNG, and other industries.

Finally, I will introduce our concept of Affordance Primitives (APs) and detail their improvements over the current Affordance Template approaches. APs have the potential to address the scalability and creation issues described above for ATs.

In summary, the primary objectives of this thesis are as follows:

- Compare the features and limitations of CRAFTSMAN and UseIt
- Integrate the compliant Cartesian jogger with UseIt
- Evaluate the degree to which integrating compliance reduces the difficulty of performing a complex task
- Introduce Affordance Primitives and how their usage relates to Affordance Templates

Below, we find that the combination of ATs/APs with the jogger alleviates operator burden while performing contact tasks and make tasks less likely to result in damage to the manipulator or manipulation object. Increased confidence in remote operation of important LNG tasks will allow trained, but not expert operators in the LNG and similar industries to deploy robots on their assets, reducing the risk of occupational hazard for their employees.

1.6 Organization

Chapter 1 introduced the problems associated with completing remote complex tasks, Affordance Templates, and how they can improve autonomy when completing complex tasks. It also describes some of the outstanding issues with ATs including how to addresses issues related to scope, contact, alignment errors, sensor noise, and scalability. Chapter 2 is a review of the relevant literature on Affordance Templates, and other task definition approaches related to ATs, as well as compliant controllers and joggers. Chapter 3 reviews and compares the two developed AT software packages in terms of the issues identified above. Chapter 4 discusses the implementation of these AT packages with the compliant Cartesian jogger. Chapter 5 introduces Affordance Primitives and summarizes how they have the potential to address the scalability issues discussed above. Chapter 6 summarizes the results and future work.

Chapter 2

Literature Review

Chapter 2 is a review of action planners, affordance templates, joggers, and compliant control. In this chapter I will not discuss motion planners as we are using MoveIt, the industry standard.

2.1 Task Definition Approaches

The first step in the process of task planning is to decide on a task representation format. Multiple tasks representation formats, including ordered lists of actions and AND/OR graphs, have been employed in robotic research. However, as Rocha points out, tasks are generally represented as a series of symbolic, high-level operations, which are then translated into motions [32]. Examples of such high-level instructions would be "pick up the ball and place it on the table" or "drive from point A to point B". These operations can then be combined to create more complex tasks, such as commanding a robot to drive to a specific location, pick up a ball, and then drive to and place the ball at a secondary location.

Nof's Robot Time and Motion (RTM) method [29] states that any robot operation can be described as a combination of the following eleven steps: move, orient, touch, grasp, release, reach, stop-on-error, stop-on-force, vision, process-time-delay,

and time-delay. The RTM method defines a large set of low level actions that can be combined into higher-level operations to define complete tasks. The low-level actions can be described in a directed graph, which define the relationship and order between each action.

Initial robotic systems employed ordered lists of actions to define tasks, which simply break down a task into a sequence of actions. Therefore, at their most basic level, ordered lists are directed graphs with only a single outgoing arc from each operation. The operations themselves typically consist of one of the eleven basic steps defined by the RTM method. These graphs are often used to define manufacturing plans [3, 43]. Ordered lists of actions are relatively inflexible as they are incapable of identifying more than a single series of actions that result in the completion of the task. However, the simplicity of ordered lists allows them to be rapidly developed [5].

Task planners have also frequently utilized AND/OR graphs to both break down complex tasks into simpler operations and to identify all feasible processes that can be generated from a set of operations [31]. AND/OR graphs reduce a high level, complex task, such as the transportation of an object, into a series of sub-goals, such as gathering the object, moving from one location to another, and placing the object at the final location. AND/OR graphs list all possible combinations and orders of sub-goals that result in the completion of the primary objective. Finally, once an AND/OR graph has been generated and all possible task execution methods identified, a task implementation method can be selected. The selection of a single task execution method generates a directed graph from the bottom AND/OR tree node to the completion of the primary goal.

As this research focuses primarily on the execution of robotic tasks on an LNG facility, at which the sequence of task steps is relatively rigidly defined due to the dangerous workplace environment, directed graphs and ordered lists of actions are employed to define tasks. Affordance templates writ large are based off of directed graphs [13], which are not as expansive as AND/OR graphs, nor as limited as an ordered list of actions.

2.2 Action Planners

To allow for a more streamlined, semi-autonomous approach to performing contact tasks, a wide variety of *action planners* have been developed and implemented. ADLs and Affordance Templates are some of the most common tools used to define and execute autonomous tasks.

Action Description Languages are a subset of action languages, which are used in the field of computer science to describe the effects of performing actions. ADLs, specifically, relate to transition systems, and are commonly used in robotics. Many ADLs exist, including the Stanford Research Institute Problem Solver (STRIPS), the Planning Domain Definition Language (PDDL), ADL-A, ADL-B, ADL-C, and several more variants.

All of these are hierarchical/conditional planning methods. They model the world as a transition system, or a set of state variables that change as the result of actions performed in the world. In STRIPS, for example, there is a set containing all possible world models based on the consequences of a set of possible actions, or "operators". Thus, each operator transforms the current world model into another

model. The objective of the problem solver is to find the string of actions that will take an initial world model and transform it into the desired final world model [7].

ADLs are mainly used for adaptive planning, plan optimization, and to reduce navigation time for performing tasks [23]. ADL-A, one of the older ADLs, was designed to "represent and reason about actions and their effects" [25]. Since the creation of the earliest Action Description Languages in 1993, there have been many offshoots of the concept, which extend to cover concurrent actions, non-deterministic effects, and indirect effects [7, 10, 23, 25]. Today, there are many niche options when choosing an Action Description Language, all with unique advantages and drawbacks for a diverse set of applications.

The development of ADLs has primarily focused on mathematical advancement and software implementation, with limited focus on ease of use. The vast majority of ADL planners, including STRIPS and PDDL, are text based, and do not lend themselves well to graphical user interfaces (GUIs). Due to the text based nature of ADL planners, significant background on ADL software syntax and terminology is required for implementation. In contrast, affordance templates, employ a GUI for alignment and task execution.

2.3 Affordance Templates

As discussed in Chapter 1, Affordance Templates are constructs that allow for the rapid implementation of pseudo-autonomous manipulation tasks. They specify EEF waypoints for a task with respect to the object to be manipulated. Then, a motion planner such as MoveIt [28] generates the trajectories between the waypoints

to execute the task motions.

2.3.1 The Theory of Affordances

Affordance Templates get their name from a popular learning theory in the field of psychology. Affordance Theory, invented by American psychologist James Jerome Gibson in 1950, states that the environment is perceived in terms of possible interactions with objects. Essentially, affordances specify the ways in which a human interacts with objects in its environment [11].

Thus, an affordance is an inherent property of an object that details the ways in which it can be used by humans or robots. Examples of affordances for everyday objects include buttons, which have an affordance for depressing, as well as knobs and handles, which have affordances for pulling, pushing, and turning. Hermans, *et al* identified the following list of affordances: pushable, rollable, graspable, liftable, dragable, carryable, and traversable [16].

While the concept of affordances is not new, it may be crucial to the way we define robot behaviour moving forward. This theory of affordances [11] is the basis for Affordance Templates, and it is a more intuitive way of thinking about robot motion and task completion than current robot-centric approaches [34]. By defining tasks in relation to the object(s) they act on, the problem of performing these tasks is no longer robot-specific, but can instead be generalized to work with any ROS-compatible robot. Additionally, since humans perceive the world in terms of affordances, it makes ATs more user-intuitive than most other methods of task planning.

2.3.2 Development

Current AT development is spearheaded by NASA and TRACLabs. Their software packages, named UseIt and CRAFTSMAN (CaRtesian-based AFfordance Template Suite for MANipulation) by NASA and TRACLabs, respectively, were designed to help partially automate the execution of complex tasks.

Both packages trace their origins to the Affordance Template software package developed by a team at NASA-JSC in 2013 [13,14]. This package was showcased in the 2013 DARPA Robotics Challenge (DRC), where it was used to supervise task behaviors for the NASA-JSC Valkyrie robot. For the DRC trials, the JSC team was using Valkyrie to turn a wheel valve, as shown in Figure 2.1. The stated goal of the DRC trials was to "develop human-supervised ground robots capable of executing complex tasks in dangerous, degraded, human-engineered environments" [1]. To demonstrate that the system met the desired goal of increased autonomy for difficult tasks, the robot was required to perform the tasks with little to no operator intervention.

In 2015, the Affordance Template development team split, with some members leaving NASA for TRACLabs. The group at TRACLabs then developed CRAFTSMAN, which utilizes a modified version of the original Affordance Templates package for task planning. The developers who remained at NASA-JSC halted Affordance Template development until 2018, where they began work on UseIt, which also builds upon the original Affordance Template package. Chapter 3 is a detailed explanation and comparison of these two software packages.

Among the precursors to UseIt and CRAFTSMAN are IHMC's Coactive De-

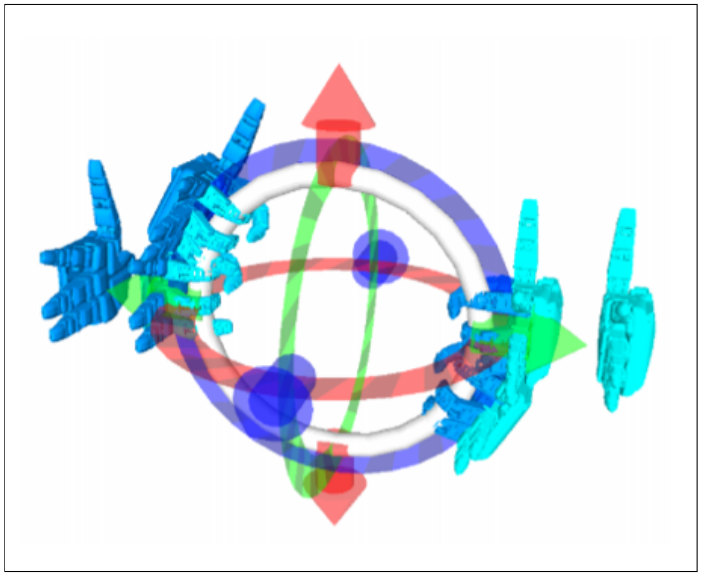


Figure 2.1: A Wheel Template Created for the Valkyrie Robot [13]

sign and MIT's Object Template Descriptive Format (OTDF) packages, both of which were also showcased at the 2013 DRC trials. These packages are similar to NASA's Affordance Template package in their intended uses and capabilities, but the AT developers were concerned that they were limited in that they could only be used with a certain few robots and they did not allow for much adjustment by users. ATs were designed with these shortcomings in mind, and developers sought to ensure that they could work with any ROS-compatible robot and would allow for a large amount of user interaction to fit diverse environments.

The coactive design methodology was devised to reach a middle ground between full autonomy and direct teleoperation. IHMC wanted robots to be useful as teammates that worked alongside humans to make them more productive and efficient rather than simply being tools humans used to achieve their goals [19]. Johnson

places an emphasis on the robotic agent understanding the effects of its actions on the environment before the agent can work as a teammate with human operators. This method is built on and expands upon Fong's Collaborative Control System Model [8] in which the human acts in a supervisory role to "supplement the robot's limited perceptual and cognitive capacity" [19].

While the DRC highly encouraged participants to design autonomous solutions to the tasks, IHMC used the coactive design methodology to create a "human-machine team" to perform the desired tasks. Some portions of the task were autonomous, such as when the robot was navigating the test area, but cognitive tasks such as recognition/identification of objects were left to the human operator. The portion of the task where the robot interacted with the environment was completed by the robot or the human, with each scenario (human-driven or robot-driven) having advantages and disadvantages. For this portion of the task, it was possible for either the operator or the robot to successfully perform the operation, but each struggled when working without the help of the other. The best performance here came when the operator was able to provide feedback to the robot to refine its autonomously-generated plan [19]. Thus, the human and robot partnership eventually evolved into supervisory control, similar to to ATs.

The Object Template Descriptive Format (OTDF) [6, 20] uses affordances as the basis of their high-level interaction scheme. They specified their template objects similarly to the AT package, whose implementation is discussed below. OTDF contained algorithms that automatically performed object fitting using a LIDAR scanner, but it also allowed for users to manually fit affordances to objects. This process reg-

istered the sensed objects to the known set of affordance objects in the affordance database, and from there the set of actions corresponding to each affordance could be performed, i.e. turning a valve or pushing a button. This differed from NASA's approach, which required operators know the task *a priori* and then import the proper model and align it manually.

Since the DRC trials in 2013, the MIT team has made several adjustments to their OTDF package, including placing a greater emphasis on their user interface component, which they named 'Director' [26]. During the trials, the MIT team's GUI was intended mostly for user feedback and not user control. Director, however, works much more like NASA's Affordance Template software package [13, 14], where the GUI was intended for human operators to have greater control and customization options. Based on the images of Director found in [26], the newer interface seems to offer a great deal of customization options to the user, but at a level that might overwhelm inexperienced operators and preclude them from being able to use the software.

2.3.3 Implementation

Current Affordance Template implementations consist of a 3D model of an object and a set of end-effector waypoints relative to that object. They are set up to contain all of the spatial data required to perform a given task, with only minor adjustments needed at run time to fit the specific environment. This makes the template highly reusable, and requires far less effort from the operator to set up and perform a task. Thus, ATs for common tasks can be set up in advance, and

merely need to be imported and aligned when it comes time to perform the task. By streamlining this process, it allows for human operators to act in a more supervisory role than directly controlling all robot motion.

The Affordance Template ROS package was designed to be layered on top of the pre-existing ROS packages for motion planning, perception, and navigation. As described in [13], the template structure is "a directed acyclic graph of display objects and ordered sequences of end effector waypoints" expressed in the frame of the task object. This structure can be seen in Figure 2.2. For the purposes of the DBB demonstration presented in Chapter 5, these ATs can essentially be simplified as ordered lists. The valve template used for the DBB contains a single valve object that includes a list of potential actions/trajectories to perform (i.e turn clockwise or counterclockwise). Each trajectory, though, is simply a single ordered list of waypoints. There is no branching within each trajectory; the different trajectories are each their own "branch" of the main template, but each trajectory is an ordered list without variation.

At run time, the template can be adjusted in terms of its position, orientation, and size. Additionally, while the template contains only one object, that object can be associated with many actions that the user might want to perform. For example, if the template object is a wheel valve, the user would likely have trajectories for turning left and right to varying degrees. In that scenario, the template might be populated with 6 trajectories for common desired actions: turn 30° clockwise, turn 60° clockwise, turn 90° clockwise, turn 30° counterclockwise, turn 60° counterclockwise, and turn 90° counterclockwise. This makes the template broad enough to perform

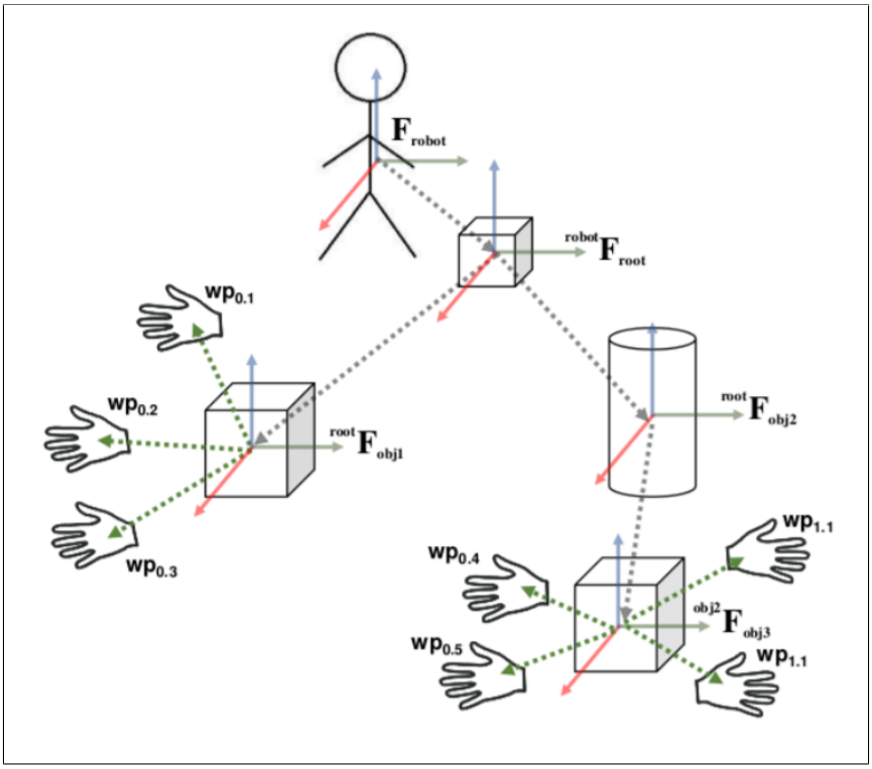


Figure 2.2: The general structure for an Affordance Template [13]

several common tasks while also limiting the solution space to avoid overwhelming the user.

Finally, waypoints may be added or removed from existing template trajectories at any time, although it is probably unwise to do so unless the template was inherently flawed. If the user desires to delete waypoints at run time due to environmental conditions or obstacles, it would be advisable to create a new, temporary AT tailored to the specific run without altering the general AT for that object/task. If the user finds that they want to delete waypoints, it is likely that the template was too large in scope and thus not generalized for wide use. As briefly mentioned in 1, in the context of ATs *scope* refers to the template's complexity. To prevent this problem, templates should contain the minimum amount of motions necessary to complete a task. Issues of scope associated with ATs are further addressed in Chapter 3.

For this project, ATs better fit the challenges of performing tasks in an LNG plant than ADLs due to the desire to reduce cognitive burden on operators and lower the skill level required to be able to perform complex tasks. Affordance Templates, with their emphasis on user-friendly implementations are much better suited to novice users than ADLs, and allow for much more user interaction. Additionally, with such a large and fluid environment as an LNG facility, it would be extremely computationally-expensive if not infeasible to set up the world model necessary for ADL implementation. By comparison, ATs are very quick to implement, and require next to no knowledge of the environment to set up.

2.3.4 Related Work

Some work has been done with affordance wayfields, which represent affordances as gradient wayfields as opposed to discrete waypoints. The wayfields are composed of "critical regions" that the EEF has to traverse to perform the desired action or actions. Constraints on motion are implemented by giving undesired regions a high cost value. Another potential advantage to using wayfields is that the desired end goal of the action can be set to a region as opposed to a particular EEF pose.

This work with wayfields is aimed at combating the most common limitations associated with Affordance Templates by allowing for control of the robot throughout the trajectory, rather than only at discrete waypoints [27]. McMahon criticized that existing approaches were often "one-off systems restricted to laboratory environments" and not well-suited for general use. He asserted that combating the limitations of existing approaches is "increasingly critical as we face the complexity of common human environments filled with uncertainty" [27].

Another interesting framework in the same vein as ATs is Object-Action Complexes (OACs) [22]. This framework works to "link robot actions to the visual and haptic perception of objects" [21]. Similarly to affordance templates, OACs are based on the belief that objects and actions are linked. This framework differs from ATs in that it relies heavily on the system learning behaviors from the environment, whereas ATs are environment-agnostic. Additionally, OACs rely on a visual perception algorithm to detect features of objects to discern their associated actions. ATs were not designed for the automatic detection of objects.

However, there is some work with affordances in the field of computer vision to automatically classify and register affordances to sensed objects [44]. Ideally, this work would be extensible to allow for the automatic generation of affordance templates for common tasks associated with an affordance, i.e. if a doorknob is sensed with the associated affordance for turning, a template can be automatically generated and populated with waypoints that will turn the knob. This would be an improvement over current affordance registration, which relies on human operators to manually register a template or on visual aids such as AR markers to facilitate alignment.

2.4 Jogging

While ATs are extremely useful for performing task motions, the user also needs a way to control the system manually. One option is to use point-to-point planning, but the user is still at the mercy of the motion planner. The method that gives the user the most control and is the most user-intuitive is jogging the robot using a joystick, motion controller, or 3D mouse.

Jogging refers to real-time control of a robot. Before joggers, motion planners only supported point-to-point control, in which the operator specifies the beginning and ending EEF positions and orientations, but the trajectory the manipulator takes between these points is determined by the motion planner. This lack of user control can lead to undesirable plans, whose only remedy is to continuously re-plan until a better trajectory is generated. To remedy this and enable the easier completion of manipulation tasks, developers sought to design a method of real-time (or nearly real-time) EEF position/velocity control that was based in ROS.

The jogger implemented in this project was first developed by Dr. Andy Zelenak, but has since become a part of MoveIt [45]. This jogger relies on velocity-based control; it receives an input of an EEF velocity and can output either joint velocities or positions:

$$\boldsymbol{\delta q}_{\text{nominal}} = \boldsymbol{J}^{\dagger}\boldsymbol{\delta x} = \boldsymbol{J}^{\dagger}\boldsymbol{K}\boldsymbol{\delta \hat{u}} \tag{2.1}$$

Where $\boldsymbol{q}_{\text{nominal}}$ is the vector of joint angles, $\boldsymbol{J}^{\dagger}$ is the pseudoinverse (discussed further below), $\boldsymbol{x}$ is the Cartesian twist, $\boldsymbol{K}$ is a diagonal matrix of positive scaling factors, and $\hat{\boldsymbol{u}}$ is the concatenated, normalized twist command.

This jogger handles singularities and allows for easy integration with an admittance or impedance controller, which sets it apart from previous real-time controllers and makes it ideal for use performing contact tasks. When a kinematic singularity is being approached, the jogger slows down and sends a warning to the user that they are approaching an undesirable configuration. This implementation uses the singular value decomposition of the Jacobian to calculate the pseudoinverse[1]:

$$\boldsymbol{J}^{\dagger} = \boldsymbol{V}\boldsymbol{S}^{-1}\boldsymbol{U}^{T} \tag{2.2}$$

Equation 2.1 is then differentiated to generate the velocity command:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\dagger}\dot{\boldsymbol{x}} = \boldsymbol{J}^{\dagger}\boldsymbol{K}\dot{\hat{\boldsymbol{u}}} \tag{2.3}$$

[1]Note that the Moore-Penrose pseudoinverse would traditionally be used in this calculation as it optimizes the manipulator's power consumption, but in this case the singular value decomposition pseudoinverse is used because it is more stable near singularities.

When approaching a singularity, the jogger reduces the EEF velocity as the condition number increases:

$$\dot{\boldsymbol{q}} = \begin{cases} \dot{\boldsymbol{q}}_{nominal}, & \kappa(\boldsymbol{J}) \leq \kappa_l \\ \dot{\boldsymbol{q}}_{nominal} * \left(1 - \frac{\kappa(\boldsymbol{J}) - \kappa_l}{\kappa_u - \kappa_l}\right), & \kappa_l < \kappa(\boldsymbol{J}) < \kappa_u \\ \boldsymbol{0}, & \kappa(\boldsymbol{J}) \geq \kappa_u \end{cases} \tag{2.4}$$

Where $\kappa(\boldsymbol{J})$ is the condition number, and κ_u and κ_l are the upper and lower condition number thresholds, respectively.

The condition number was calculated using the Eigen C++ linear algebra library:

$$\kappa(\boldsymbol{J}) \approx \left\|\boldsymbol{J}^{\dagger}\right\|_F \|\boldsymbol{J}\|_F \approx \frac{\sigma_{\max}(\boldsymbol{J})}{\sigma_{\min}(\boldsymbol{J})} \tag{2.5}$$

This method of slowing when approaching singularities and not allowing users to drive the manipulator into poor configurations was designed in part to aid non-expert users. As the robot is not allowed to get too close to the singularity, the user is always able to jog the arm back into a more desirable configuration. Additionally, while the user might not understand what a singularity is or why it is undesirable, they intuitively know that the arm should not be slowing down, so they likely realize that the near-singular configuration is poor [45].

As a final safety feature, the jogger runs a collision check thread that ensures that the manipulator does not collide with itself. The features detailed above make this jogger safe and friendly for users at all levels of experience. Further details on this jogger can be found in [45].

2.5 Compliance

As discussed in Chapter 1, compliance refers to the flexibility introduced to a system via either passive or active measures. The advantages of increased compliance of robotic systems made it a popular research area beginning in the 80s [17, 24, 35].

Passive compliance comes from hardware elements such as series elastic actuators, which contain springs that lower mechanical impedance and friction. Prime examples of a compliant or "soft" robot are the HEBI robotics systems [15]. Unfortunately, compliant robots sacrifice both precision and power to increase their elasticity over traditional stiff, industrial manipulators. This makes them less attractive for use in commercial industries. Ideally, industries want to implement systems that can maximize safety by finding a balance between precision and compliance.

Active compliance is primarily used for stiff manipulators (though it can be used with passively compliant manipulators as well). This type of compliance relies on the controller to set a virtual impedance that acts much in the same way as the spring an a series elastic actuator. This method was originally proposed by Neville Hogan in 1984, who suggested that impedance control would simplify manipulator control [17].

Generally, active compliance works by introducing a virtual impedance between the robot and the environment. For the implementation described in Chapter 4, the compliance controller is calculating the impedance joint velocities due to contact forces on the EEF at the same time that the jogger is calculating the joint velocities for the desired EEF movement. An impedance control law is used to calculate the

complaint EEF velocity $\dot{\mathbf{x}}_{\text{comp}}$:

$$\dot{\mathbf{x}}_{\text{comp}} = \mathbf{K}^{-1}\left(\xi - \xi_{apply}\right) - \mathbf{B}^{-1}\dot{\xi} \tag{2.6}$$

Where $\mathbf{K}$ and $\mathbf{B}$ are diagonal matrices for stiffness and damping, ξ is the external applied wrench, and ξ_{apply} is the desired wrench applied to the environment.

The compliance and the jogging command are combined to produce the resultant joint velocities:

$$\dot{\mathbf{q}}_{\text{total}} = J^{\dagger}\,\text{diag}(\nu)\dot{\mathbf{x}}_{\mathbf{jog}} + J^{\dagger}\,\text{diag}(\mu)\dot{\mathbf{x}}_{\text{comp}} \tag{2.7}$$

Where $\dot{\mathbf{x}}_{\text{comp}}$ is the compliance calculated in 2.6 and $\dot{\mathbf{x}}_{\text{jog}}$ is the user input, or the jogging command.

An advantage of active compliance over passive compliance is that the stiffness of the system can be changed. Whereas the mechanical impedance of a passively compliant robot is set, since the controller handles impedance it can be changed at any point during operation to fit changing environmental conditions or varying needs throughout a task. This allows for stiff manipulators to become compliant, but requires precise sensor data for forces and torques experienced by the EEF.

The compliance controller used in this project was also developed in our lab for use with our stiff manipulators. It was primarily developed using UR robots which, in the absence of an external force torque sensor, must calculate the EEF forces and torques based on the current draw at each joint. This proved to be very noisy data, and the first iteration of the compliance controller was thus prone to error. With

the addition of a force torque sensor installed at the EEF, the compliance controller performs much more reliably.

Several improvements have also been made to the controller that allow for compliance to be further specified to meet the user's needs:

- The user can choose to activate compliance for only certain dimensions
- The user may set a different stiffness for each compliant dimension
- The user man change compliance parameters mid-task
- A GUI was created to allow the user to control and keep track of compliant dimensions

By allowing rigid manipulators to act like passively complaint systems, a compliant controller like the one described above will automatically manage contact forces. This will allow for more novice user to perform contact tasks without the fear of exceeding safety limits. Additionally, the customization afforded by the improvements listed above allow for compliance to be as dynamic as the tasks being performed. The ability to change the compliance parameters mid-task increases the versatility of the system to be able to handle even more difficult or complex tasks than was previously possible.

2.6 Summary

While there are several options when it comes to task planning, Affordance Templates present the best option for use in hazardous industrial environments. Older

approaches such as Coactive Design and the Object Template Descriptive Format were similar in structure to ATs, but did not allow for users to make necessary adjustments to their templates. The more recent development of affordance wayfields provides users with much greater control over task motions, but this level of control is likely far too detailed for novice users. ATs strike the perfect balance. Their level of adjustability is such that a single template can be employed in a wide variety of situations and environmental conditions, but not so complex that they require dedicated roboticists to build and use them.

Most of the robots that are precise enough or capable of applying sufficient force in industrial and/or field applications are rigid manipulators, and thus not intrinsically compliant when in contact with their environment. Without the aid of a compliant controller, the operator must closely monitor the force with which the EEF is contacting objects in the environment so as not to damage the manipulator or the task object. When a compliant controller is implemented, the controller removes this burden from the operator.

Layering the compliant controller on top of affordance templates has the potential to allow the operator more leeway when aligning the template to the task object. Additionally, compliance parameters can be changed during the task to best fit the needs of each motion being performed. Together, affordance templates with compliance should alleviate the cognitive burden on the user and reduce the risk of task failure due to misalignment from human error or noisy sensor data. The reduction in task difficulty then enables more novice users to run capable robotic systems and perform tasks remotely.

Chapter 3

Affordance Template Package Comparison

A portion of this research project is focused on evaluating the performance of two Affordance Template packages, CRAFTSMAN and UseIt, and comparing their AT capabilities. The ideal Affordance Template package will balance usability, power, and flexibility for use in numerous real-world applications. This analysis serves as a starting point for research groups and corporations to identify the optimal package for AT usage in their industry or field. Additionally, the Affordance Template execution procedure for each software is defined, providing novice users with a starting point for initial testing in UseIt or CRAFTSMAN.

3.1 The Ideal Affordance Template Software

Before introducing the two Affordance Template packages evaluated in this research, it is important to discuss the desirable attributes of a general AT software package. The primary goal of implementing Affordance Templates is to improve the automation of complex tasks and eliminate the need for remote manual control. This has several benefits including reducing operator burden, making the task more repeatable, and freeing up employees to multitask.

To achieve their intended purpose, ATs must be able to perform a wide variety

of tasks, from simple navigation and surveillance tasks to more complex manipulation tasks. The breakdown of tasks will vary depending on the industry and company utilizing the robots, but the ideal AT package would have the capability to perform most common contact tasks, including picking up and moving small objects, pressing buttons, turning valves and wheels, and opening doors. In more complex scenarios, they may be used to interact with non-rigid objects such as unzipping and inspecting a bag after it is removed from a vehicle.

Additionally, ATs need to cater to operators with wide levels of experience. If the AT is too difficult for the average employee to use, it is less desirable for industries to implement robots in their operations because it would come with the added cost of having dedicated, highly-trained AT operators on hand to use them. It is much better if any employee can use the robot or robots to complete tasks with minimal training. With ATs, the operator should have to do little more than supervise the task completion, which is much less demanding of the employee's time and attention.

It is extremely important that a general Affordance Template package balances advanced capabilities with a user-intuitive interface. This is critical because it is easy for a user interface to become overwhelming if it is built to be used by someone with expertise in the field. A GUI designed with expert operators in mind would allow for more powerful and precise control, but would exclude any non-experts from using the product unless the GUI properly displays only highly important options in the main section, while restricting more advanced options in separate menus.

Finally, the ideal AT package must use industry standard, open-source packages for planning and inverse kinematics. In this case, that means that the ATs

should be built upon motion planners and kinematics solvers that are trusted and widely-used. This helps for several reasons: these packages are often easier to install, have a great deal of documentation for solving common issues, and have a significant user base that decreases the likelihood of glitches or errors in the program.

With these metrics in mind, CRAFTSMAN and UseIt are evaluated and a recommendation given on the use cases for both software packages.

3.2 History

As mentioned in Chapter 2, CRAFTSMAN and UseIt trace their origins back to the same package, NASA's Affordance Template ROS package. However, not long after the 2013 DARPA Robotics Challenge the developers of the original software package parted ways and began independent development on the two separate packages described in this chapter. As such, the packages are very similar in their core structure, but have followed different development paths, leading to different capabilities and focusing on different user bases/use cases.

Of the two packages, CRAFTSMAN has seen a greater development effort and is the more mature product. CRAFTSMAN was created by TRACLabs to remedy the lack of an industry standard open-source application for robust, user-friendly task and motion planning. While applications for motion and task planning already existed, these packages were either proprietary, limited to specific hardware, too complex for non-expert users, or tailored to a specific lab environment [38]. TRACLabs wanted CRAFTSMAN to be versatile and robust enough to be useful for real world applications.

CRAFTSMAN is much larger than just its Affordance Template package. In addition to ATs, the CRAFTSMAN software suite also contains libraries for inverse kinematics, Cartesian motion planning, graphical teleoperation, and finite state machine design [37]. To ensure user-friendliness, the GUIs for these packages are all built as Rviz plugins. Having all of these packages in a single software suite allows for the easy integration of all of the components necessary to perform a complex contact task and also provides a consistent and intuitive user interface.

UseIt is the newer AT package that is being developed by the Robonaut 2 team at NASA's Johnson Space Center in Houston, Texas. Unlike CRAFTSMAN, UseIt is a stand-alone Affordance Template package, and not part of a larger software suite. UseIt was born when the team resurrected the original Affordance Template ROS package after a gap of several years as a portion of their collaborations with Woodside. They have since added several new capabilities and re-branded the package as UseIt, but the underlying structure remains similar to the original Affordance Template package. Recently, UseIt has received more concentrated development, and UseIt 2.0 is on the way.

UseIt 2.0 will feature several quality-of-life improvements over UseIt 1.0. First, the world will be able to contain multiple models and sub-models, allowing for templates to better fit more complex, multi-object tasks. Additionally, the new model structure will incorporate key-frames, which are frames set at any interest point in the template (e.g. the desired contact point on a task object). Models will also be given "state" information, such as a valve or door being open or closed, or some degree between the two. Finally, UseIt 2.0 will replace the terms "Robot" and "Trajectory"

with “Agent” and “Action Path” to better fit the wide variety of autonomous or semi-autonomous systems capable of performing tasks. Combined, these changes will allow make UseIt 2.0 a more powerful tool for task automation.

3.3 Craftsman

As the more mature product, CRAFTSMAN is evaluated first. CRAFTSMAN defines spatial tasks in its AT library in one of two ways: the templates can be created ahead of time and saved as .json files or they can be created and saved at run-time using the RVizCraftsmanPanel, which is shown in figure 3.1 and explained in more detail in the section below. The ability to create templates using the Rviz GUI allows operators with less robotics experience to create and execute Affordance Templates to complete difficult tasks.

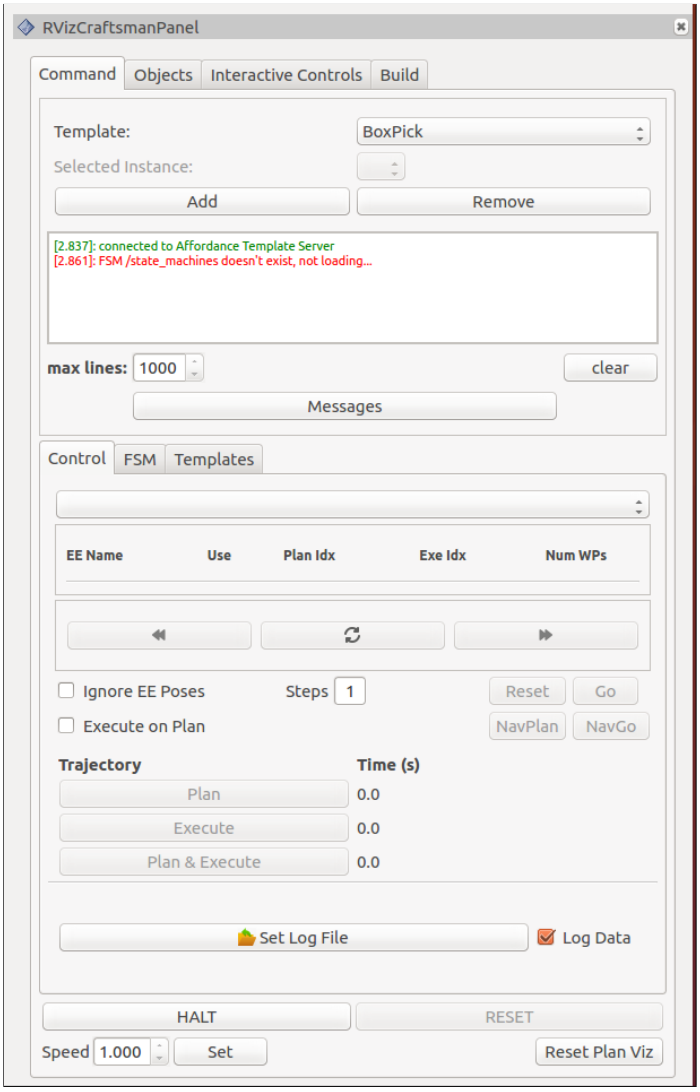


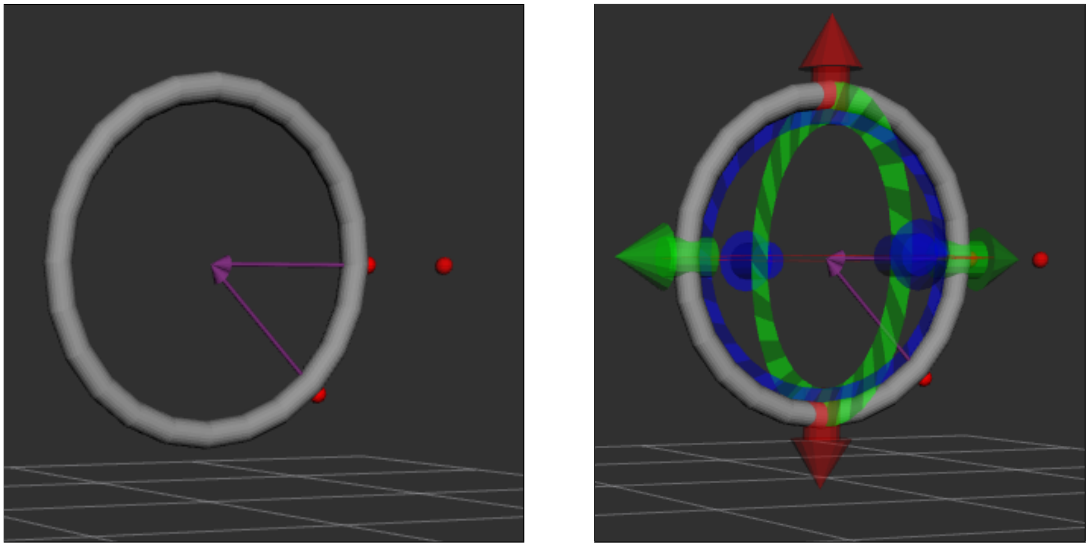
Figure 3.1: RvizCraftsmanPanel

This section will detail how CRAFTSMAN's ATs work, how to use CRAFTSMAN to create and execute ATs, and some of CRAFTSMAN's notable features and limitations.

3.3.1 Using Craftsman

To load a template into Rviz using CRAFTSMAN, the user must select the desired template from the selection menu in the top of the RVizCraftsmanPanel and click "Add". After loading the template into Rviz, the user must first move the template model to align it with its real-world counterpart. To move the model, the operator must right-click on the model and select "Hide Controls" to deselect that option and reveal the model's interactive marker [12]. This marker allows the user to drag or rotate the template about its three principal axes to align the template appropriately. Figure 3.3b shows an example of a template with the interactive marker visible.

If the task object is not already within reach of the manipulator(s), the user will also need to add or adjust existing navigation waypoints to position the robot close enough to the task object to comfortably reach throughout template execution. Navigation waypoints are added by right-clicking on the blue cylinder at the origin point of the mobile base and selecting "Add Waypoint After" under "Advanced". Existing navigation waypoints can initially only be moved in the X-Y plane by dragging the cylinder or rotated about the Z axis by dragging the blue ring at the base of the cylinder. The interactive marker can be brought up by right-clicking the cylinder and selecting "Toggle Full Control" under "Advanced". As seen in Figures 3.2 and 3.3 below, CRAFTSMAN displays the waypoints for navigation as upright transparent cylinders and the waypoints for manipulation as solid red spheres, with arrows to the center of any rotation point.



(a) Without Interactive Marker (b) With Interactive Marker

Figure 3.3: CRAFTSMAN's Wheel Template

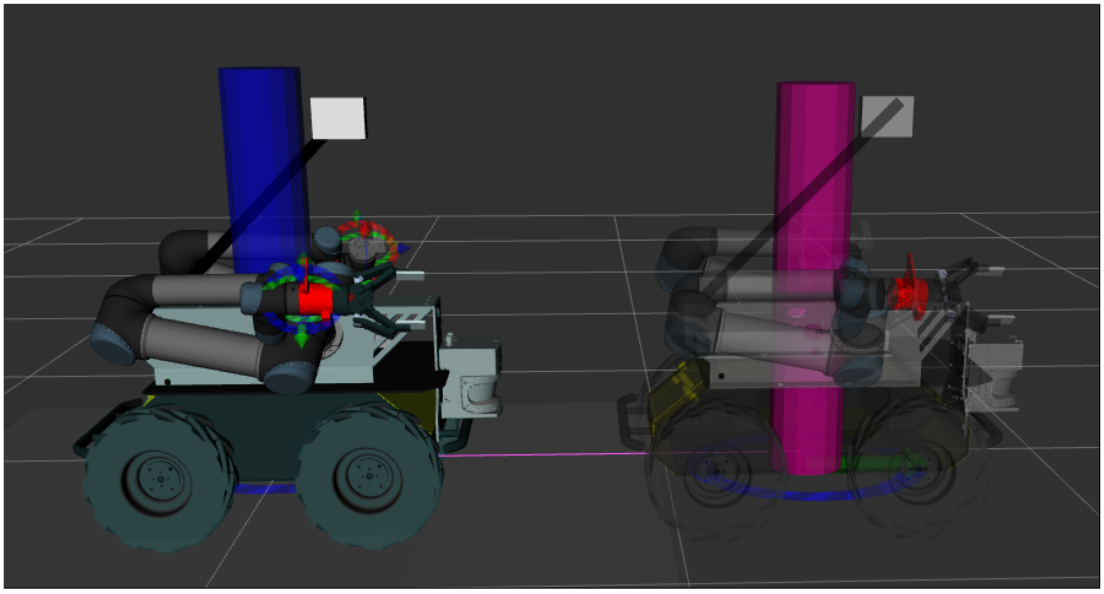


Figure 3.2: CRAFTSMAN Navigation

Now that the template is aligned and the navigation waypoints are set, the user may execute the template. To navigate, the user simply selects the goal cylinder (there can be more than one navigation waypoint) and clicks "NavPlan" on the right side of the RVizCraftsmanPanel under the control tab. The trajectory for navigation is then displayed as a purple spline in Rviz and the phantom robot shows what the execution of the plan will look like by following that path. If the plan is correct, the user clicks "NavGo" to begin driving.

For manipulation, the user first selects one of the template trajectory options, shown in Figure 3.4, which can be selected in the RvizCraftsmanPanel or can be selected within the Rviz window by right-clicking on the template model. For the wheel template, there are three options: turn the valve using the left arm, right arm, or both arms. After selecting the desired trajectory, the user clicks "Plan" in the RVizCraftsmanPanel and the trajectory for the arm/arms is displayed in green in Rviz, as shown in Figure 3.5, and the phantom robot executes the plan. If the plan is correct, the user clicks "Execute" to execute the trajectory in full, which is a significant advancement over the original Affordance Template ROS package, where the user had to plan between each waypoint and the next. CRAFTSMAN also allows the user to select the "Plan & Execute" option if they wish to do the options in a single step.

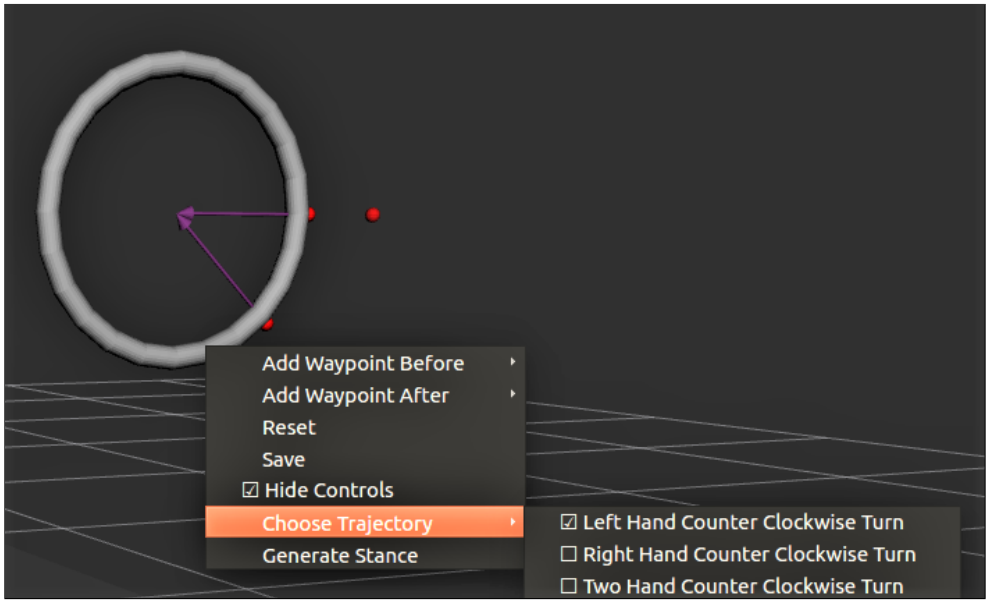
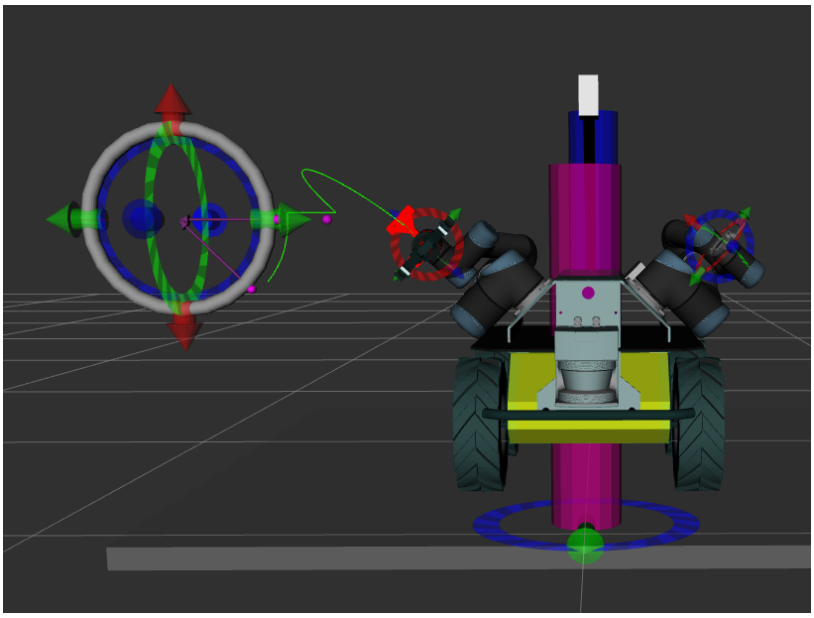


Figure 3.4: Trajectory Options for CRAFTSMAN's Wheel Template

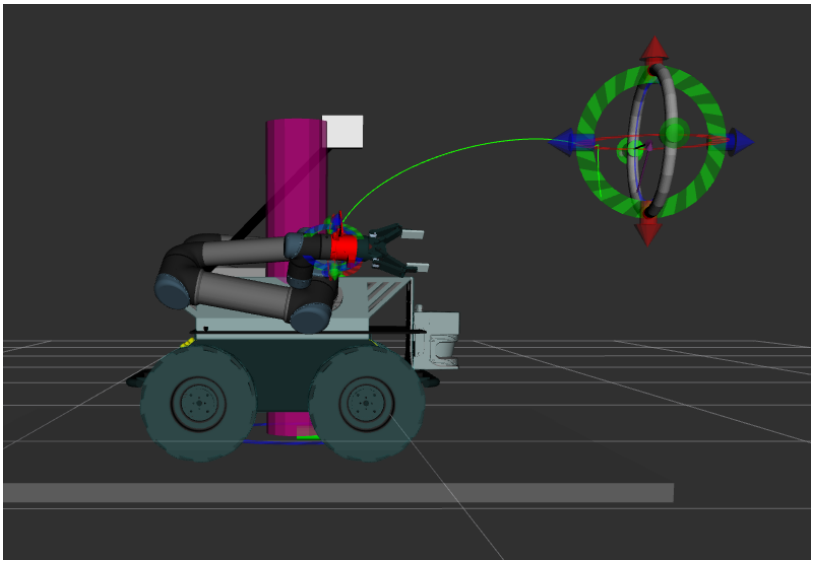
To create a template from scratch, CRAFTSMAN has implemented a helpful template creation wizard to aid the user in generating a new template. This wizard is launched when the user opens the "Build" tab at the top of the RvizCraftsmanPanel and clicks "Start".

The first page of the wizard asks the user to name the template and select the folder in the file system where the template will be stored. Next, the user must add at least one display object to the template. The wizard prompts the user to name the object, choose the display type (primitive or mesh), and – if there are multiple objects – select the parent object. If the user selects mesh, they must input the path to the mesh file and set the scaling for the object.

If the user selects primitive, they must set the primitive type (box, cylinder,



(a) Frontal View



(b) Side View

Figure 3.5: Manipulation with CRAFTSMAN's AT Package

or sphere), the size (x, y, and z), and the color (red, yellow, blue, green, or magenta). They must also go through the added step of setting the display controls for the primitive, as shown in Figure 3.6. This includes setting the pose and orientation (x, y, z, roll, pitch, and yaw), the control dimensions, and the control scale.

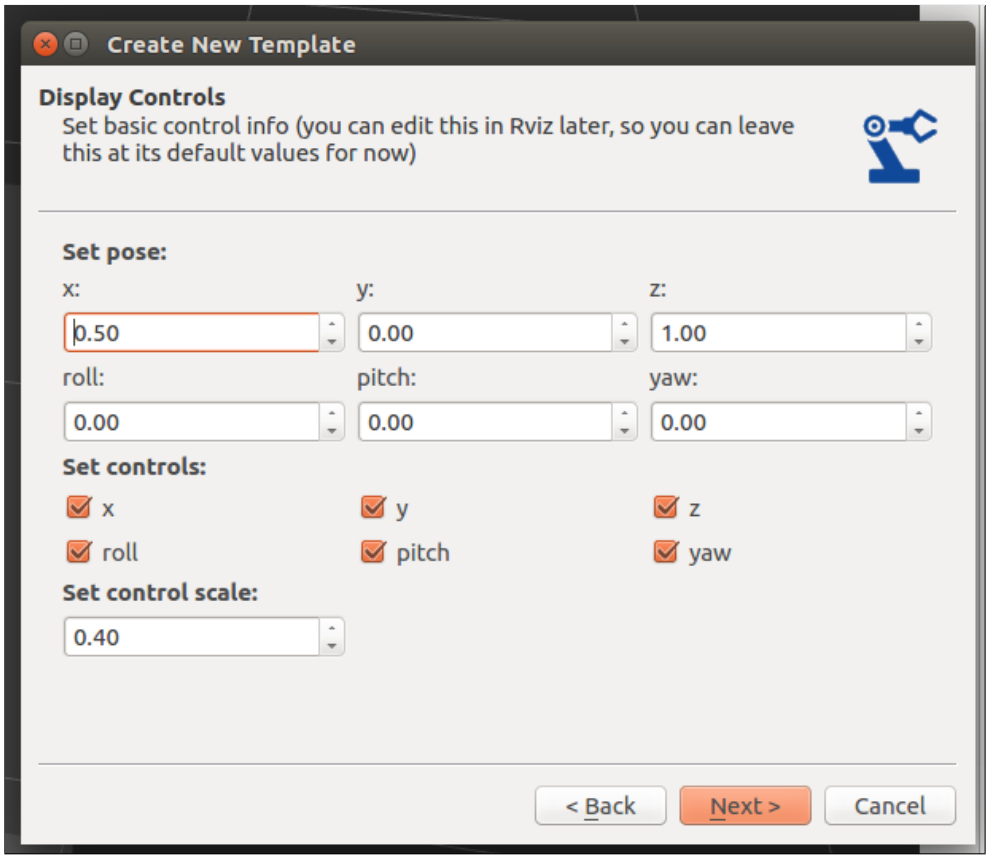


Figure 3.6: Template Display Controls

Once the object is added, the user must set the trajectory name and select which end effectors the trajectory will use. Finally, the user adds a waypoint, as shown in Figure 3.7. Here, the user selects the EEF "pose" (open, closed, half-closed,

or mostly closed), template origin (x, y, z, roll, pitch, and yaw), control dimensions, control scale, tool offset, tolerances, metric (minimum distance, manipulability product, or manipulability ratio), and plan type (joint or Cartesian).

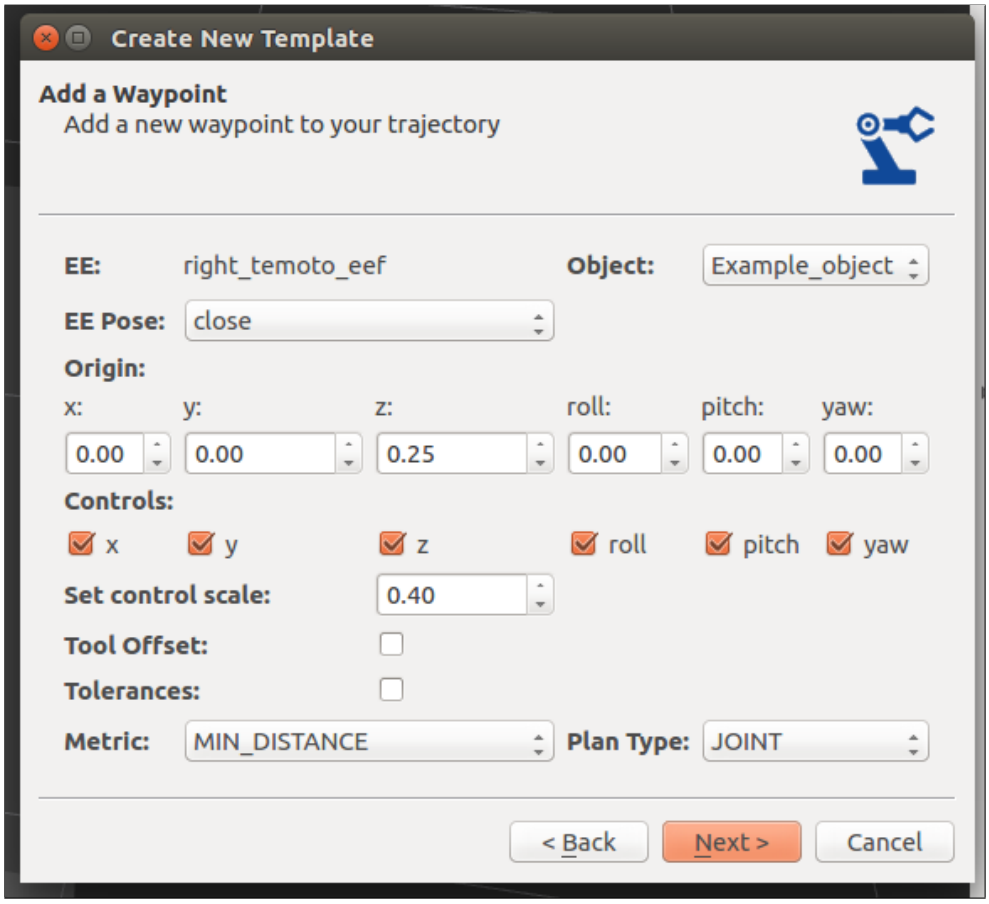


Figure 3.7: Adding a Template Waypoint

After adding the first waypoint, the wizard closes and the user finishes editing the template in the Rviz window. To add more manipulation waypoints, the user right-clicks on the first waypoint and selects either "Add Waypoint Before" or "Add Waypoint After" and selects the desired EEF. The waypoint then appears in the

Rviz window as a red sphere with a visible interactive marker, and the user may drag this waypoint to the appropriate location. Once placed, the user can right-click and select "Hide Controls" to hide the interactive marker. The waypoint cannot be moved (relative to the object model) while the controls are hidden, meaning there is no risk of accidentally moving a waypoint when clicking in the Rviz window. Navigation waypoints can also be added in the manner described at the beginning of this section.

3.3.2 Features and Limitations

One of CRAFTSMAN's main features is its motion planning software. Instead of using MoveIt's default inverse kinematics (IK) solver Orocos Kinematics and Dynamics Library (KDL), CRAFTSMAN uses trac_ik for motion planning. TRACLabs' developers noticed that they faced a large number of solve errors when using KDL as their IK solver in the 2013 DARPA Robotics Challenge [**?**]. They determined that imposed joint limits were causing KDL to experience these errors [2, 39]. TRACLabs created trac_ik as an alternative to KDL that performs better when used with joint-limited robots. Results from testing IK solvers using a variety of kinematics chains showed that trac_ik had an average solve rate over 99% whereas KDL's solve rate varied from 45% to 93% with an average of 77% [2].

Along with the increased reliability of trac_ik over KDL, CRAFTSMAN also displays generated EEF trajectories as a green spline in Rviz. Normally, the trajectory's execution is only shown by a phantom robot, which can be played a single time or on a loop. CRAFTSMAN displays the trajectory as a permanent spline in Rviz that does not vanish unless the operator re-plans (in which case the new trajectory

is then displayed) or the operator executes the trajectory. This permanent trajectory visualization makes it easier for the user to confirm that the plan will not violate any constraints or collide with the environment as they can rotate the Rviz window to view the trajectory from multiple angles without having to wait for the planning animation to loop.

Additionally, CRAFTSMAN generates a comprehensive trajectory that plans through all of the waypoints in one motion as opposed to generating a series of small trajectories between each waypoint and the next. This means that the entire task is performed in one smooth motion and the user only needs to press "Plan" and "Execute", or "Plan & Execute", once to run the template in full. Combined with the full trajectory being shown as a green spline in the Rviz window, this greatly reduces the time to perform the task as the user does not have to repeatedly plan and execute between each set of waypoints to check that each portion of the plan is valid. The trajectory spline allows the user to simply plan once, verify that the planner has generated a good plan, and then execute the full template. This reduces the template execution to a single user command and saves time that would be wasted by repeated planning.

Another feature of CRAFTSMAN is that once a template has been loaded into Rviz, the user can perform most template actions by right-clicking in the Rviz window. This allows the user to minimize the RvizCraftsmanPanel, whose multiple tabs might be overwhelming to novice users, and enlarge the viewing area.

CRAFTSMAN's primary limitation is that its GUI can be overwhelming without any training. The advanced capabilities of this package mean that there are far

more options for the user to choose from, which results in initial work in CRAFTSMAN being significantly slowed by time spent searching for a specific option or capability. However, while the initial ramp-up time to become familiar with CRAFTSMAN can be significant, the CRAFTSMAN user interface with Rviz is well designed with a series of effective features. For example, the operator can click on the end effectors and drag them around, then they can then simply right-click the EEF and a menu appears in the Rviz window to plan and execute to that location. It's very simple, allows for more rapid planning, and avoids having to open an additional window. Again, the primary drawback of this feature, and the CRAFTSMAN user interface as a whole, is that the feature(s) can be difficult to identify due to the large amounts of tabs on each GUI panel. To overcome this limitation in a corporate environment, specialists would be required to spend additional time training and assisting novice operators.

3.4 UseIt

UseIt is the newer Affordance Template package, originally designed for use with Robonaut in space. Being partially funded by Woodside, whose goal is for everyday employees to use robots to perform complex tasks on their LNG facilities, a central goal of UseIt's development was to ensure that minimal robotics knowledge is necessary to use Affordance Templates. This emphasis on user-friendliness during the early stages of development has led to different design decisions and prioritized features.

3.4.1 Using UseIt

UseIt is a standalone Affordance Template package that takes advantage of Rviz's interactive markers in the same way that CRAFTSMAN does, but utilizes its own separate GUI, shown in Figure 3.8, instead of integrating the GUI as an Rviz plugin. Additionally, UseIt saves its templates in XML format as opposed to JSON, and new templates are primarily created and saved using the UseIt GUI.

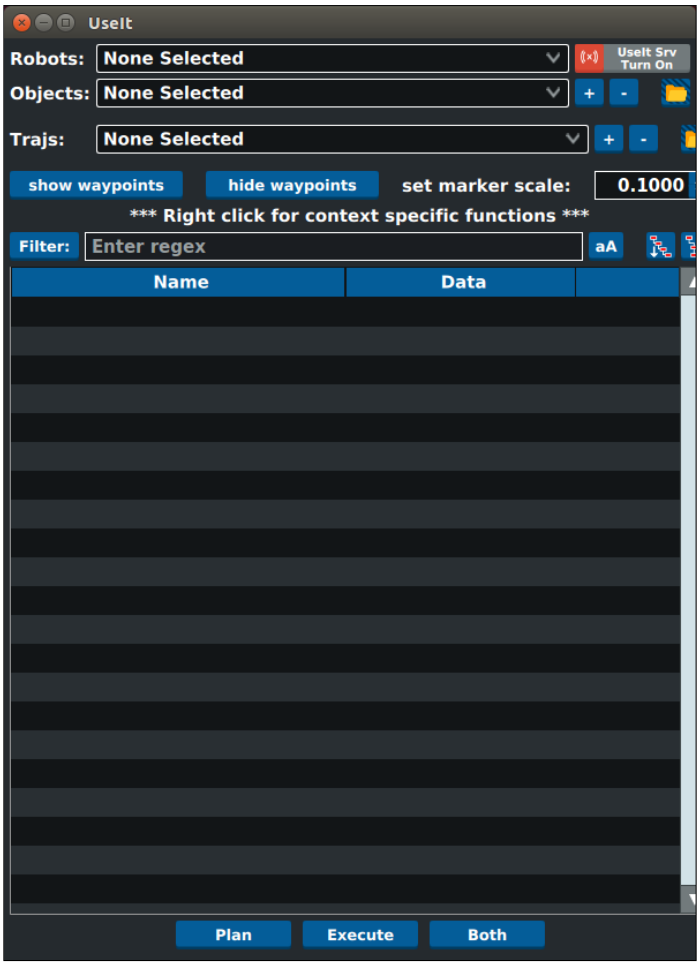


Figure 3.8: The UseIt GUI

UseIt's GUI is more user-intuitive, with only three selection menus on the initial screen as opposed to CRAFTSMAN's seven tabs and two menus, with more options within each of those tabs. In total, UseIt has 15 API capabilities CRAFTSMAN has 38.

The UseIt GUI is set up in such a way that the process of loading an existing template is very straightforward. First, the user selects the robot from the "Robots" selection menu, then they choose the template model they wish to use from the "Objects" menu. Finally, they select the template trajectory or action that they wish to perform on the task object from the "Trajs" menu.

The template is loaded into Rviz with the interactive marker visible, so the user can choose to freely drag the template into the correct position, or they can drag along a particular axis to move the model. Each waypoint also shows as a set of axes with its own interactive marker, which can be toggled on or off by selecting the "hide waypoints" box in the GUI. Using the point cloud data, the user aligns the template model to the real-world object.

Once the model has been aligned, the user simply steps through the template by pressing "Plan" and "Execute" for each individual waypoint in the trajectory sequence. As shown in Figure 3.9, the waypoint name turns yellow while planning or executing, green if the plan or execution succeeds, and red if the plan or execution fails. In the future, users should be able to press a single button to execute the template as a whole instead of stepping through by planning and executing to each individual waypoint. To ensure that no wild plans are automatically executed once this feature is added, the user will likely have to impose joint limits or other constraints

on the motion planner. If any of the generated plans between consecutive waypoints fails, the template could be set to either try to re-plan a set number of times or until success, or it could be set to exit the template altogether.

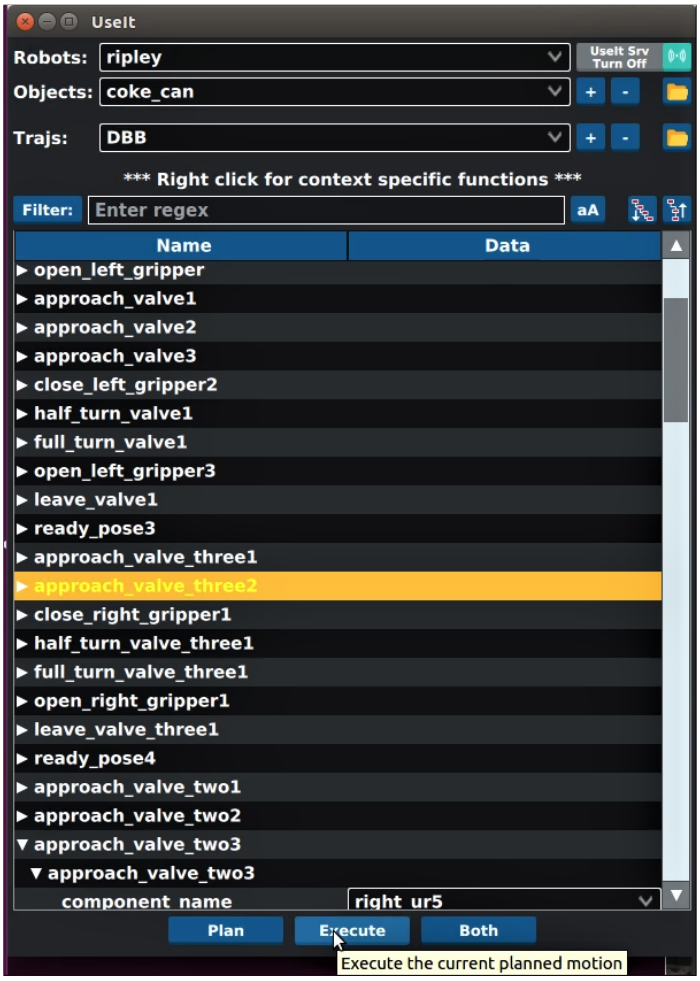


Figure 3.9: Stepping Through a Template

To build a new template, the user follows the same initial steps as loading an existing template. First, they select the robot from the menu in the GUI. Next, they

can either select an existing object model from the selection menu, or they can add their own by clicking the "+" to the right of the "Objects" selection menu. This brings up a dialog box to add a new or existing object. The user selects the "New" tab and then must enter the object name, the ROS package containing the mesh, the path to the mesh file, and the scaling factors, and then clicks "Add". The folder icons to the far right of the "Objects" and "Trajs" menus allow the user to save the object or trajectory they have created.

Next, the user will follow a similar step as above and click the "+" to the right of the "Trajs" menu. This brings up a similar dialog box to adding a new object. The user again selects the "New" tab and is asked to enter a name for the new trajectory. Once the name is selected and the user clicks "Add", they may begin populating the new trajectory with waypoints. The user needs to remember to save their progress frequently by clicking on the folder icon to the right of the "Trajs" menu.

To generate the waypoints, the user right-clicks in the waypoint list area – the striped area that makes up the bottom two-thirds of the GUI. The following waypoint options appear:

- Plan to Waypoint
- Execute to Waypoint
- Plan And Execute to Waypoint
- Insert Waypoint Before
- Insert Waypoint After

- Remove Waypoint
- Add State

Initially, the waypoint list is empty, and the user must select "Insert Waypoint Before" (or "Insert Waypoint After") to add the first waypoint. A dialog box will appear where the user may enter the waypoint name. The user should enter a unique name for each unique waypoint to avoid confusion. Once the name is entered, the user must right click on the waypoint. The same waypoint options detailed above will appear again. This time, the user should select "Add State". Another dialog box will appear, and the user should enter the state name, which should be the same as the waypoint name. In addition to entering the name, the user must choose the state type from the following list of options[1]:

- Cylinder Grasp Interaction
- Joint
- Named Pose State
- Pose
- Robotiq2F85 Gripper

[1]These waypoint options are specific to the robot that UseIt was developed on and the tasks the project was developed for (namely, valve-turning and button-pushing). The three most common options are explained in grater detail below. Additional state options may be added in the future to fit a wider variety of tasks and grippers.

Of the state options listed above, "Named Pose State" is the simplest. It populates the waypoint with pose data from any named pose in the robot's SRDF. Once this option is selected, the user must expand the waypoint and state information by clicking on the arrow to the left of the waypoint and state names. Now the user can see the Named Pose State options, as shown in Figure 3.10. The "component_name" selection menu contains all of the motion planning groups defined in the SRDF. The user selects whichever planning group is associated with the desired named pose. Though there are entry fields for several other options, the only other applicable field for the Named Pose State is the bottom field, labeled "pose_name". Here the user must type out the name of the desired pose, making sure to check that the spelling and capitalization match the SRDF. This waypoint is now fully-functional, and the user may plan and execute to the named pose.

Figure 3.10: Named Pose State Options

For the "Pose" state option, the operator follows the same procedure as with the Named Pose State through the component selection step. After that, the user has two options for populating the pose data: they can either manually enter the calculated desired pose in to the available fields, or they can plan or jog the arm to the desired position and populate the pose data from the current robot pose. (The latter is done by right-clicking on the state - nested beneath the waypoint - and selecting "Populate Pose From Robot", as shown in Figure 3.11.)

Figure 3.11: Populate Pose from Robot

The "Robotiq2F85 Gripper" state option is specific to the hardware used. UseIt was developed using a Universal Robots UR5 arm, with a Robotiq 2-Finger 85 Adaptive Robot Gripper. As with the options for the other state types, the "Robotiq2F85 Gripper" state requires the user to select the component, set the axes scale, set the gripper command (which controls the degree of open-ness), and set the

ROS topics for the gripper. While this state type is specific to the Robotiq gripper, future gripper state types will need to include the same general information to be able to close and open new grippers using UseIt.

As demonstrated in the section above, UseIt is easy to use whether simply running an existing template or building one from scratch. Its GUI is straightforward and avoids overwhelming the user or over-complicating the template creation or execution process. Users of any skill level will be able to use UseIt with a minimal amount of training.

3.4.2 Features and Limitations

One of *UseIt*'s newest features is the "Populate Pose From Robot" option described above. The ability to set a template waypoint based on the current pose and orientation of the end effector is convenient for manipulators with a free-drive or "teach" mode where users can move the end effector of the arm by hand to the desired position and orientation[2]. Thus, if the manipulator is posed in front of the task object, the time to set up the template can be greatly reduced as the user can quickly and precisely move the EEF to the desired waypoint locations. Additionally, this simplifies the template generation process to the point where a novice user could easily produce their own template in a few minutes. Being able to physically control the manipulator additionally reduces the cognitive burden on the user, who does not have to use a controller or worry about input commands. Of course, not all robots

[2]It is important to note that joint angles are not preserved in UseIt, only the end effector position and orientation are saved from this action; the overall arm configuration is not captured.

have this teach capability, so the feature is hardware agnostic.

The largest limitation of UseIt is that it is not yet able to be used for navigation tasks. This limits the ability of its Affordance Templates to manipulation and surveillance tasks. For the sake of performing contact tasks this is not a large problem, but it does leave the burden of positioning the robot within ample reach of the task object to the operator. Due to remote driving experience provided to many adults by RC cars and video games, navigation tasks are perhaps the least burdensome on the operator of the three main task categories (manipulation, navigation, and surveillance). However, this is time that the operator will have to devote their full attention to this task that could otherwise be spent performing other duties while merely supervising the AT.

3.5 Comparative Summary

To show a more direct comparison, Table 3.1 lists some important metrics of CRAFTSMAN and UseIt.

Table 3.1: Software Comparison Overview

	CRAFTSMAN	**UseIt**
IK Solver	trac_ik	KDL[3]
Navigation	Y	N
Active Development	Y	Y
Dual-Arm Plans	Y	N[4]
Number of Exposed API Capabilities	38	15
Number of Primitive Shapes	3	N/A[5]
Ease of Use (1-5)	3	2
Learning Curve (1-5)	4	1

3.6 Concluding Remarks

Affordance Templates are a task execution framework that aims to increase automation of a wide variety of tasks. This chapter described the ideal AT package and introduced two existing packages, CRAFTSMAN and UseIt. Both packages were implemented in simulation in the same version of Ubuntu with the same robot. The processes of executing existing templates and generating new templates were described for each package. Finally, the notable features and limitations for each

[3]UseIt uses whichever solver the client selects when setting up their robot with MoveIt.
[4]UseIt is not set up to plan with both arms unless it is using the Named Pose State option.
[5]UseIt requires the use of mesh objects.

package were discussed and a recommendation was given for the best use-case for each package.

Additionally, this chapter discussed the scope of affordance templates, which is an important aspect that has yet to be strictly defined. Does a template for opening a door contain just the information to turn the knob or does it contain all of the information to perform the task, including positioning the robot in front of the door, turning the knob, pushing the door open, and driving through? Based on the evaluation of both software packages detailed above, the scope of the UseIt Affordance Templates is currently limited to encompass only small actions, i.e. turning a knob. However, CRAFTSMAN, being the more mature product and having already incorporated navigation into its template capabilities, shows the potential for the scope of its ATs to be more all-encompassing. Therefore, CRAFTSMAN is optimal for more complex tasks, while UseIt, with a simpler graphical user interface and more limited command options, allows novice users to more easily complete tasks using Affordance Templates.

As briefly mentioned in Chapter 1, an aspect of Affordance Templates that needs to be refined moving forward is their scope. In their original form, Affordance Templates were extremely limited in scope so as to make them as generalized as possible. For example, a valve would have two "actions", turn clockwise and turn counterclockwise. These templates would be as sparsely populated with waypoints as was possible to still allow for successful task completion. As the scope of an AT grows, it becomes tailored to a specific task configuration and is no longer as versatile. This is a disadvantage because it means that the user needs to create more templates,

which somewhat defeats their intention.

Using CRAFTSMAN, ATs can encompass an entire task from start to finish. CRAFTSMAN's enhanced scope over other Affordance Templates makes it an ideal candidate to perform complex LNG processes. The ability to add navigation waypoints to ATs further eliminates responsibilities that traditionally fall on the operator.

Conversely, it can be desirable to leave affordance templates as small in scope as possible to perform a basic task. Then, complex tasks are essentially "macro-templates" built from the combination of several small templates run in quick succession. For example, for a double block and bleed procedure the operator could add three separate templates in to the planning scene, one for each valve. If the valves are all of the same type, the same template would be used for all three of the valves, just loaded in separately. Then, the macro-template would specify which action to take for each template instance as well as the order in which to execute the actions. By keeping the templates small in scope, their reusability is preserved without making the task difficult for the operator to set up.

Chapter 4

Affordance Template & Compliant Controller Integration

Chapter 4 discusses the integration of Affordance Templates with the compliant controller. The results are demonstrated completing a double block and bleed valve turning task in Perth, Western Australia from a remote computer in Austin, Texas. Controlling the robot from such a great distance introduces additional burden on the operator in the form of latency between sending commands and receiving updated sensor data. The addition of compliance to ATs helps offset that burden and reduce the risk of task failure caused by an improperly aligned template.

This project began in 2017 when a representative from Woodside visited the NRG lab in Austin and decided to purchase a nearly-identical copy of our dual-arm mobile manipulator, Vaultbot. Woodside purchased this system for use on their LNG facilities in Western Australia and employed the Nuclear and Applied Robotics Group to help develop manipulation capabilities for this system. Vaultbot's sister robot in Perth was named "Ripley", and these two robots, shown in figure 4.1, were used as the main development and testing platforms for this research.

(a) UT Austin's "Vaultbot"

(b) Woodside's "Ripley"

Figure 4.1: The twin robots

4.1 Affordance Template and Compliant Controller Integration

The first step in performing any complex contact task using compliant affordance templates is to ensure the arm is correctly set up to run whichever motion planning package the AT uses to plan between its waypoints. At the minimum, the motion planning software should support point-to-point planning with collision checking, but jogging is also a useful feature. Point-to-point planning is used during template execution to plan between the waypoints, and either jogging or point-to-point planning can be used when generating the template depending on user preference. The AT software must also work with the gripper to send open and close commands.

From a hardware perspective, the robot must be outfitted with the proper sensor suite to facilitate the use of ATs and compliance. ATs require the robot to be equipped with some sort of LiDAR or depth camera to align the virtual template to

its real-world location. The compliant controller requires the manipulator to have a force torque sensor. Some manipulators may have F/T sensing capabilities built into them, but many require an external F/T sensor to be added on.

Next, the template must be generated in advance by an experienced user. It is not necessary to have a copy of the task object(s) on hand when creating the template, but it does help to be able to test the template locally and verify its accuracy during template setup. During this process, the operator may use either jogging or point-to-point planning to move to the desired waypoint locations relative to the task object and record the end effector pose and orientation.

Finally, the optimal compliance parameters and dimensions for the task must be identified. This process can be more difficult as the parameters are highly specific to both the task and the manipulator. These parameters may be identified through extensive user testing in which the user performs the task via jogging with compliance.

The general procedure detailed in this section was employed for the DBB demonstration described in the remainder of this chapter.

4.2 Implementation Details

4.2.1 Hardware

Vaulbot, whose configuration was developed at UT Austin in 2012, is a Clearpath Robotics Husky mobile base with two Universal Robots UR5 manipulators. Vaultbot's base is also outfitted with a SICK LMS511-20100 LiDAR and two Kodak PixPro SP360 panoramic cameras. The dual manipulators allow for a broader range of motion and a greater reach than a single manipulator. The mobile base with attached

LiDAR is ideal for navigating in near-flat environments and identifying obstacles. Finally, the camera feeds from the panoramic cameras are stitched together in RViz to create a video sphere, shown in Figure 4.2, that provides the operator with a full 360 degrees of vision while performing navigation tasks[1].

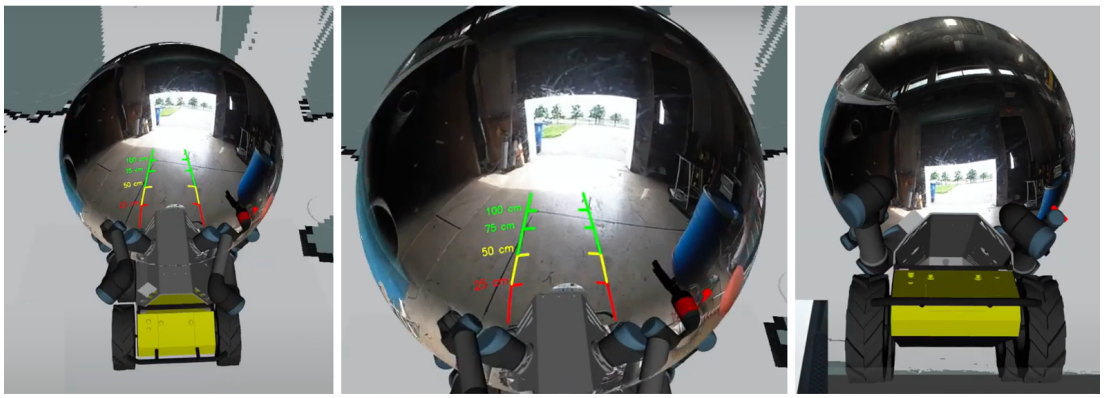


Figure 4.2: Vaultbot's 360-degree Sphere in Rviz

For this particular project, one of Vaultbot's UR5 arms was equipped with a Robotiq 2-Finger 140 Adaptive Robot Gripper, an ATI Gamma force torque sensor, and an Intel RealSense D435 depth camera. These additions to the UR5 allow for enhanced user awareness and for the integration of a compliant controller, which relies on the feedback from the F/T sensor. While Vaultbot only had compliance active for one of its UR5 arms during this demonstration, it is possible to enable compliance on multiple arms if each is equipped with a gripper and F/T sensor. A full description of Vaultbot's hardware can be found in Table 4.1.

Vaulbot was used for the development of manipulation capabilities at UT,

[1]The Rviz plugin for the video sphere can be found on the NRG's public github page, at https://github.com/UTNuclearRoboticsPublic/rviz_textured_sphere

Table 4.1: Vaultbot's Hardware Components

Component	**Manufacturer**	**Model**
Mobile Base	Clearpath Robotics	Husky
Arms	Universal Robots	UR5
Depth Cameras	Intel	RealSense D435
Panoramic Cameras	Kodak	PixPro SP360
F/T Sensor	ATI	Gamma
Grippers	Robotiq	2-Finger 140
Lidar	SICK	LMS511-20100

and Woodside's Ripley was used to demonstrate the software by performing a remote demonstration at Woodside's robotics lab in Perth. Similar to Chen and Trivedi performing testing in multiple locations to ensure their system architecture was "flexible, general, and robust" [4], this effort took advantage of the existence of the twin robots on opposite sides of the globe to test the compliant affordance template structure remotely. The demonstration proved the effectiveness of the compliant AT approach and the robustness of the software to work on multiple systems with different hardware.

For the final demonstration, Ripley had several minor hardware differences from Vaultbot. First, Ripley was outfitted with two Robotiq 85 grippers, one on each EEF, whereas Vaulbot had only one, larger gripper. Additionally, each of Ripley's grippers had a Robotiq F/T sensor attached to it, allowing for compliant manipulation using both arms instead of just one. Finally, both arms were equipped with RealSense depth cameras, allowing for even greater situational awareness as one arm could act as a dynamic/mobile camera while the other was used for manipulation. Figure 4.3

shows a screenshot of the RealSense camera feeds being used in the manner described above. These changes combined to make Ripley an excellent system for executing complex manipulation tasks.

Figure 4.3: Dual RealSense Camera Feeds

4.3 Additional Software

For the remote teleoperation demonstration, UseIt was the AT package being used. Since UseIt does not yet support navigation tasks, the navigation portion of the task had to be performed using either point-to-point control or jogging. Similarly, the final portion of the demonstration called for the user to generate a new AT on the fly, which also required the user to manually control the EEF into the correct waypoint positions. For these reasons, we employed TeMoto [40, 41] for the portions of the task where ATs alone could not be used.

TeMoto - Japanese for "at hand" - is a software package for intuitive teleoperation. Developed by the NRG, TeMoto was designed to manage commands from a

variety of user-friendly input devices, from controllers to human speech. Figures 4.4 and 4.5 show the commands for two of the many input devices TeMoto supports, a 3Dconnexion SpaceMouse and an HTC Vive hand-controller[2]. TeMoto interfaces with MoveIt! and thus supports both point-to-point planning as well as jogging, depending on the user's preferred control input method or combination of methods. Finally, TeMoto is also used to send gripper open/close messages to the manipulator and to bias, enable, or disable compliance. TeMoto is structured around using a controller in conjunction with Rviz' interactive markers, so it does not have its own GUI.

Figure 4.4: Button mappings for the VR hand controllers used with TeMoto

[2]VR hand controllers can be used for jogging, but do not lend themselves well to point-to-point control as they have very few buttons to map commands to.

Figure 4.5: Button mappings for the Spacenav Pro controller used with TeMoto

4.4 User Interface

For this demonstration, comprehensive situational awareness aids were necessary to compensate for the large lag from Perth to Austin. These overlays, shown in Figure 4.7, were used to provide the operator with enough "real time" information to compensate for this lag. Figure 4.7a shows the distance marker overlaid on the ground in front of the robot. Figure 4.7b shows the rotation of the EEF, the input lag and strength of the input command, and the forces on the EEF.

Without these overlays, it was often unclear if the user's input command had been received by the remote system. With the addition of the overlays, the strength and duration of the commands being received by the robot were clearly visible to the operator.

In addition to the command overlays, the user was equipped with a simple GUI, shown in 4.6, for compliance and directional control. This GUI was fairly crude and intended only as a stopgap measure until a more thoughtful design can be implemented. Before the creation of this GUI, the only method of enabling directional control or directional compliance was by sending commands via the command line. Implementing the compliance GUI simplified the process of sending compliance commands to the robot.

Figure 4.6: The Temporary Compliance GUI

4.5 The Task

As mentioned in Chapter 1, the aim of this research is to simplify the execution of contact tasks to make robotic systems a more viable solution for a wide variety

(a) Distance marker overlay (b) Rotation, velocity, and force overlay

Figure 4.7: The Situational Awareness Overlays

of industries. This project, specifically, is funded by Woodside to enable the use of mobile manipulators on their LNG facilities. These robots will be used primarily for "3D" contact tasks - those deemed dull, dirty, or dangerous - as well as for emergency response.

4.5.1 Task Description

The representative task chosen here to test this compliant AT implementation is the double block and bleed procedure. The DBB apparatus used for this demonstration is shown in Figure 4.8. The DBB is an excellent representative task to demonstrate this research as it is a fairly common task performed in the oil and gas industry and it requires the manipulator to execute relatively complex motions that, if not done precisely, are likely to exceed safe force limits. Employing ATs with compliance should greatly reduce the completion time over the traditional teleoperation approach of jogging. In addition to saving time, the compliance controller

should eliminate the risk of exceeding force/torque limits and causing a safety fault, where the low-level arm controller shuts off the arm once the force limit is exceeded to prevent the arm from causing damage to itself or the environment.

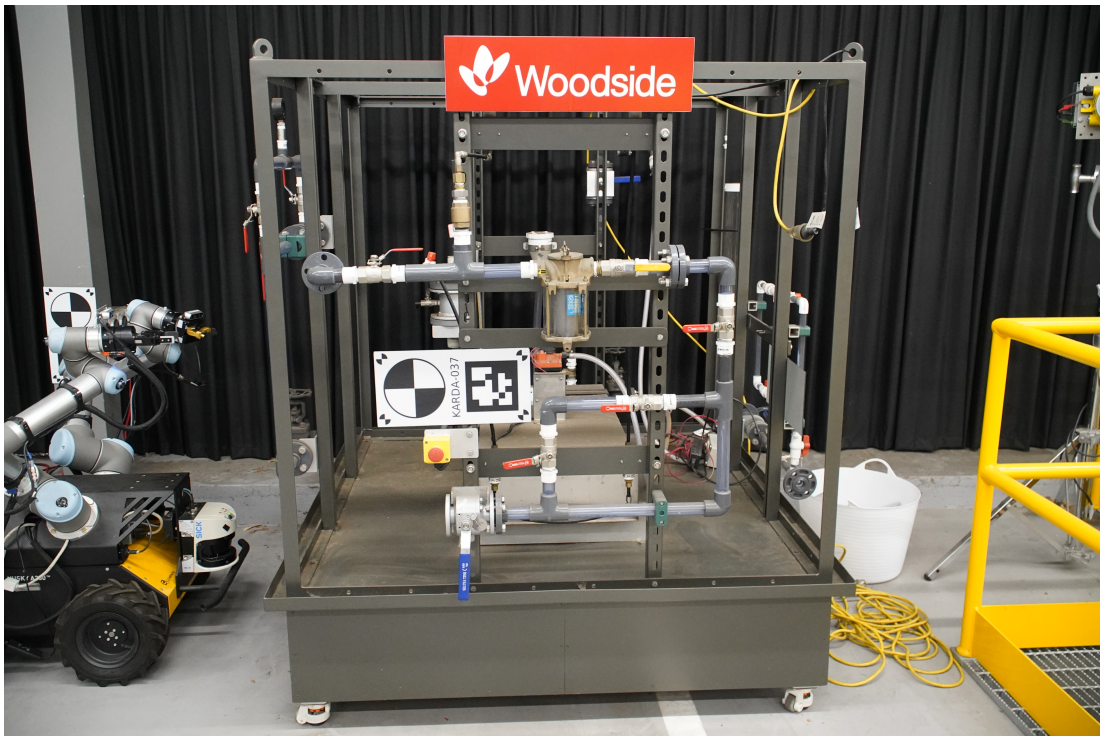


Figure 4.8: Woodside's DBB test apparatus with AR marker

The remote demonstration was performed with members of the Woodside team supervising and controlling the emergency stop, but they were not allowed to provide the remote operator with any feedback during task execution. Woodside also had a professional camera crew on hand to record the demonstration. This video is not yet public record.

This demonstration was designed to showcase a wide range of Ripley's capa-

bilities, focusing on but not limited to UseIt and compliance. The operator was to utilize both planning and jogging for navigation and manipulation portions of the demo. Thus, the steps for the remote demonstration were as follows:

- Using an input device of their choice, the operator must drive the robot to the task location and position the robot within reach of the three valves
- Using the jogger without compliance, the operator must press the process shutoff button and subsequently turn a valve
- Using the jogger with compliance, the operator must press the process shutoff button and subsequently turn a valve
- Using Rviz, the operator must localize the valve apparatus based on an AR (augmented reality) marker
- The operator must use a pre-made Affordance Template to perform a double block and bleed procedure using both arms without compliance
- The operator must use a pre-made Affordance Template to perform a double block and bleed procedure using both arms with compliance
- Using the UseIt GUI and MoveIt, the operator must create, and then run, a simple AT to press the button and turn a single valve with compliance
- The operator must stow the arms and drive the robot back to its charge station

4.5.2 Challenges

The primary challenges associated with this demonstration include latency and data volume, which led to poor situational awareness for the user. These challenges are not unique to the DBB, but rather must be addressed for any complex contact tasks performed remotely in uncertain environments.

During initial remote testing, the 250 millisecond latency[3] posed a problem for connecting to the move group action severs, preventing commands from being sent to the robot. There was a timeout parameter that had to be adjusted to allow for a greater number of connection attempts over a longer period of time. After adjusting the MoveIt timeout parameter, it was possible to connect to the move group action servers and send commands to Ripley, but the amount of time it took to launch Rviz and TeMoto was still inconveniently large. In total, it took nearly ten minutes to launch all of the necessary programs.

To remedy the slow start-up time, the Woodside team implemented a multimaster setup, which reduces bandwidth by selectively subscribing to a subset of topics. After implementing this multimaster setup, the launch time was reduced from ten minutes to less than one minute. A multimaster approach was also implemented at UT to minimize the camera lag[4], which was present when running locally over Ethernet as well as when running remotely over wi-fi, though to a lesser extent. Another advantage to the multimaster setup is that one computer and its associated

[3] 250 ms was the nominal ping during testing, with occasional spikes up to 350 ms.

[4] The multimaster implementation at UT caused some issues with the ability to run and/or connect to a VPN, but Woodside did not run into the same roadblock.

processes maybe restarted without affecting the other computer and its processes.

While operating Ripley, there was a significant amount of data passed from Ripley to the control computer in Austin. This data was necessary for situational awareness purposes, but the bandwidth was large enough to cause significant delay to the data streams, also causing them to lose their synchronization. These data streams included four video streams: the two panospheric Kodak cameras, which were stitched into a sphere in Rviz to aid the operator while driving, and the two RealSense cameras, which were mounted on the end of each manipulator to show the viewpoint of the end effectors.

Latency for the camera feeds was several seconds, and the feeds did not update simultaneously, which made it difficult to determine when the robot had stopped its motion. This latency did slow the execution of the task slightly, but the situational awareness aids employed helped combat this issue.

4.6 Results

The results of the demonstration are discussed separately for each of the four trial cases: jogging without compliance, jogging with compliance, affordance templates without compliance, and affordance with compliance. Each of these trials was evaluated based on the success of the task, time to complete the task, and the number of safety fault events. Finally, a template was generated from the remote computer and executed as a proof of concept for remote template generation.

This demonstration is intended to showcase the effectiveness of ATs in combi-

nation with a compliant controller to reduce manipulation task difficulty. Each trial run was performed only once with the same individual operator performing all four iterations of the task. It is understood that these results do not represent a statistically significant user study, but rather a successful case of software prototyping, with additional testing necessary.

4.6.1 Jogging Without Compliance

The first iteration of the remote demonstration required the operator to successfully perform a button press and a subsequent valve turn via jogging without compliance enabled. Without compliance active, it was difficult (even for someone who is quite experienced performing the valve turn with TeMoto) to turn the valve. With the large latency in the video feeds and the limited angles with which to view the valve, it was nearly impossible to determine the speed or force of the EEF in the environment. To enhance user situational awareness and make the task a bit easier, the left arm was first jogged to point the camera at the first valve to be turned. Then, the operator was able to jog the right arm to turn the valve. Having an isotropic view of the valve allowed the operator to better comprehend how close they were to the valve when it came time to grasp it.

On the first attempt at turning the valve, the operator had to settle for closing the gripper and simply nudging the valve closed. Even with the simplified task strategy, there were still multiple security fault events. While the this approach might work for small ball valves with low internal friction, it is unlikely that this shortcut would work in most scenarios.

In total, it took 28:15 or 1,695 seconds just to "turn" the first valve. For this iteration of the task, the video feed latency exacerbated the lack of compliance, causing the task to be nearly impossible to complete due to repeatedly safety faulting the arm. Over the course of this run, the operator accumulated seven safety fault events and was unable to progress past the first valve.

Thus, this run counted as a failure as only one of the three valves was turned. Further attempts to turn the subsequent valves could have been made, but the length of time required to complete the entire DBB was prohibitive. For comparison's sake, the results for the remaining three iterations will be directly compared to the results of the first, incomplete iteration. Thus, the comparison will only include the data for the run through the completion of the first valve turn. For the iterations the task was completed in its entirety, data will additionally be provided for the run as a whole.

4.6.2 Jogging with Compliance

For the second iteration of the task, compliance was enabled while jogging in an attempt to help mitigate the negative effects caused by the video feed latency. The compliance parameters for this iteration were:

- Stiffness: [500, 500, 1000, 100, 100, 5] , with the first three elements in N/(m/s) and the final three elements in N*m/(rad/s)
- Damping: [10000, 10000, 10000, 10000, 10000, 10000], with the first three elements in N/m and the final three elements in N*m/rad
- Max Force: 45 N

- Max Torque: 45 N*m
- Binary Default Compliant Dimensions: [1, 1, 0, 1, 1, 1], which represent [trans_x, trans_y, trans_z, rot_x, rot_y, rot_z]

These parameters were kept the same for the compliant jogging and compliant AT runs.

While jogging with compliance was undeniably more efficient than jogging without compliance, the difficulty of the task was still pronounced. Due to the large bandwidth of the sensor data that had to be relayed to the operator to provide adequate situational awareness for manual control, there is a significant lag between the operator's commanded input and the sensor feedback. This makes it difficult for inexperienced operators to gauge the "strength" of their commands. Furthermore, the added compliance only marginally reduced the burden on the operator as they were still employing manual control over the manipulator, which requires uninterrupted focus.

For this iteration of the test, enabling compliance led to a reduction in the completion time for the task as well as a reduction in the number of errors due to exceeding force limits. The time to complete the button press and the first valve turn was 10:14, or 614 seconds, and there were no errors during the run. This was a reduction in both time and number of errors, showing that implementing the compliance controller had a positive effect on the outcome of the task. Despite the aid of compliance, the task was still prohibitively difficult to complete in its entirety, so this iteration of the DBB task was also deemed a failure due to incompletion.

4.6.3 Affordance Templates without Compliance

For the third iteration of the task, the control method switched from jogging to using an Affordance Template to perform the DBB, and compliance was disabled again. Without compliance, the AT needed to be precisely aligned, or else the EEF was likely to exceed the force limit set by the low-level controller and cause the arm to safety fault[5]. Therefore, the operator must spend more time carefully aligning the template to the sensor data to avoid losing even more time having to reset the robot.

Table 4.2 describes the AT used in this demonstration, from the beginning of the DBB through the first valve turn.

For this trial, the switch from jogging to using an Affordance Template led to a large reduction in the completion time for the task due to the template being pre-programmed with the task motions. The time to complete the first valve turn was 4:31, or 271 seconds, and there were zero errors during the run. This was a reduction in both time and number of errors, showing that Affordance Templates are a more efficient method of task execution than manual control. The setup time for the template was 3:34 or 214 seconds, and the time to complete the full DBB task was 13:40, or 820 seconds.

4.6.4 Affordance Templates with Compliance

For the fourth iteration of the task, the DBB template was performed using an AT with compliance enabled. This time, the template was able to perform with no

[5]Note that while this means that the operator must reset the arm and the task to the beginning, the low-level controller will stop before the task object is harmed.

Table 4.2: Partial Affordance Template for the DBB Task

AT Step	**Waypoint Name**
1	Ready Pose
2	Close Left Gripper
3	Approach E-Stop
4	Press E-Stop
5	Leave E-Stop
6	Open Left Gripper
7	Ready Pose 2
8	Approach First Valve
9	Close Left Gripper
10	Half-Turn First Valve
11	Full-Turn First Valve
12	Open Left Gripper
13	Leave First Valve
14	Ready Pose 3

faults, despite intentional template misalignment by the operator. Proper compliance parameters enabled the EEF to be moderately misaligned without causing the task to fail.

In this run, the total time for the operator to perform the first valve turn was 05:29, or 329 seconds, with zero safety faults. Of that time, 2:26, or 146 seconds, was spent re-planning because MoveIt was either failing to generate a plan or generating bad plans that involved large arm motions that would have collided with the environment. Had the template executed without any time wasted due to MoveIt plan failures, the total time to complete the task would have been merely 03:03, or 183

seconds, which would have been similar to the time for the non-compliant AT run. The time to complete the full DBB was 13:38 or 818 seconds, which is almost identical to that of the non-compliant run, despite the 146 seconds of re-planning time.

The time to align the template for this run was only 44 seconds, a significant reduction from the non-compliant run's 214 seconds. The operator was comfortable with a looser alignment because of the confidence in the compliant controller's ability to manage the forces on the EEF caused by the misalignment. Thus, the compliant controller, when used with Affordance Templates saves time during task setup by reducing the required alignment/set up time. It also saves time during task execution by preventing safety faults, which waste time by requiring the operator to reset the arm's low-level controllers. In general, the time taken to execute each step of the template should not vary unless the planner is producing bad plans.

While there was seemingly little change in the execution time or number of security faults with compliance versus without compliance for the AT runs, that fails to consider the intentional misalignment of the template. Without compliance, the operator was careful when aligning the template in an effort to reduce the chance of error. When compliance wan enabled, the opposite was true: the operator intentionally misaligned the EEF (with respect to both rotation and translation) to show the added robustness afforded by the compliance controller. Therefore, while the two runs seem similar at first glance, the added compliance allows the operator to have much greater confidence that the system can execute the task when conditions are not perfect.

4.6.5 Populating a Template Remotely

In addition to the four iterations of the remote demonstration described above, the user was tasked with generating an affordance template on the fly from their remote computer. This template only required the robot to press the button and turn a single valve, not to perform the entire DBB. Generating the template from scratch from a remote computer was tedious. The entire process of template setup took 37:14, or 2,234 seconds, and the operator caused one safety fault during that time. Conversely, after it was set up the new template successfully completed the task in 6:13, or 373 seconds, with zero safety faults.

Similar to the jogging methods listed above, the EEF that was not being used for manipulation was pointed towards the task object to provide an isotropic view. The operator then used their choice of either point-to-point planning or jogging to move the EEF to desirable waypoints. At each waypoint, the UseIt GUI was used to populate the waypoint from the current robot pose. In total, the generated template contained 15 waypoints, six to press the button and nine to turn the valve. Since MoveIt provides no control over the trajectory between waypoints, the operator used a greater number of waypoints to avoid any large motions that might lead to troublesome plans. If an undesirable plan is generated during the execution of the AT, the user's only options are to re-plan until MoveIt generates a different plan, or to reset the task and slightly move the template in Rviz. With compliance enabled, the slight realignment should not affect the ability of the template to run, but the shift might cause the planner to find a new, better solution.

While remote template generation would be a niche use-case, there are in-

stances where it would be desirable to create a template for a remote application. For example, if there were no expert roboticists located at a facility where a new AT needs to be implemented rather quickly, it would make the most sense for an expert operator to teleoperate the system and generate the template remotely. Another instance would be if the robot is sent into a hazardous environment to perform a highly specialized task; it would be unlikely for the template to already exist and the operator would not be able to command the robot locally as access to the area would be restricted.

4.7 Summary

The test case detailed in this chapter with a user in Austin, Texas operating a robot 10,000 miles away in Perth, Western Australia is an extreme scenario that highlights the flaws inherent in high-latency teleoperation. Having the operator control the robot from such a great distance exaggerates the burden on the operator and demonstrates the advantages of the compliant controller combined with affordance templates even more clearly than in normal circumstances.

The full double block and bleed procedure was considered a failure when jogging. While it was possible to complete the entire DBB procedure while jogging, it is not feasible to complete the task in a reasonable amount of time for LNG facility operations. Additionally, jogging for such long periods places a large cognitive strain on the user. Despite the compliance controller managing the forces, the user must still manually plan each EEF motion, both translations and compatible rotations, and must rely on lagging visual feedback while doing so. This makes multi-step tasks

extremely tedious and inefficient.

The demonstration highlighted the benefits of Affordance Templates over traditional teleoperation (i.e. jogging) as well as the benefits of adding compliance to a rigid system. The inclusion of Affordance Templates greatly reduced the DBB task execution time, while the use of compliant control reduced the quantity of safety faults that occurred during task execution and reduced setup time by allowing for sloppier alignments. This illustrates how the combination of AT with compliance ease operator burden, presenting a path for novice users to complete complex contact tasks in dangerous industrial environments.

Although the scenario of an operator being stationed such a great distance from the system they wish to operate is unlikely (except in the case of NASA's Robonaut, which resides on the ISS), it is beneficial to test our implementation in these harsher-than-life conditions. If we can succeed in performing a task when conditions are worse than they are ever expected to be in reality, we can ensure that our approach is intuitive and easy to use in all reasonable circumstances.

Table 4.3: Remote Demonstration Results

Iteration	**Success**	**Time (s)**	**Errors**
Case 1: Jogging without Compliance	No (33%)	1695	7
Case 2: Jogging with Compliance	N (33%)	614	0
Case 3: Affordance Templates without Compliance	Yes	271	0
Case 4: Affordance Templates with Compliance	Yes	329	0
Task 1: Remote Template Generation (1 valve only)	Yes*	2,234	1

The results of the demonstration show that Affordance Templates and compli-

ance each individually work to lessen the difficulty of performing a contact task, with an even greater effect when they are used together, as summarized in Table 4.3. The time necessary to complete the task was reduced with each iteration, with jogging sans compliance being the slowest method. While the ATs without compliance run was slightly faster than the ATs with compliance run, the latter was faster and less burdensome once task set-up time was taken into account. In addition to reducing task execution time, the inclusion of compliance eliminated the risk of the EEFs exceeding safe force and torque limits, meaning that there were no safety faults during task execution. Finally we include the time to generate a remote template, but this primarily informational and not a case that can be compared to other tests.

In this implementation, compliance parameters were set independently of the affordance template, but the task at hand greatly influenced parameter selection. To make this implementation more smooth, compliance needs to become integrated with the templates during creation and not left up to the operator at run-time. A template with compliance built-in when it is produced has the potential to be used by a much wider range of operators and implemented more quickly when it is time to perform a task.

Chapter 5

Affordance Primitives

Chapter 5 introduces our concept of affordance primitives and details their advantages over traditional Affordance Templates[1]. Affordance Primitives build upon ATs and attempt to solve some of their shortcomings, while also creating a clear path for future growth and development.

5.1 Motivation

Affordance templates are highly effective for the pseudo-automation of complex contact tasks, as demonstrated in Chapter 4. However, the biggest shortcoming of Affordance Templates is their inability to manage contact forces during contact task execution. To remedy this shortcoming, the Nuclear and Applied Robotics group has developed Affordance Primitives, which integrate Affordance Templates and compliance into a single entity. By integrating compliance parameters within Affordance Templates during their creation, Affordance Primitives can be simplified to the minimum set of simple motions that will achieve a given task. Additionally, these primitives allow us to instantiate Affordance Templates with compliance parameters

[1]Chapter 5 is partially based on a previous publication [30] in which Adam Pettinger contributed the compliance control capability which is utilized by the Affordance Primitives that are the focus of this effort.

for each task or sub- task, reducing the complexity of each portion of a task.

For example, we consider using the jogger to turn a valve or a handle. Without compliance, the operator must jog the arm through a sweeping arc motion while also commanding the rotation of the gripper at a compatible rate so that the EEF force limit is not exceeded. If the user is not careful, it is easy to rotate the EEF faster or slower than the rate of translation and this over- or under-rotation can cause a safety fault. However, if the EEF is freely compliant in the roll dimension, the minimum necessary user input can be simplified to a straight line tangent to the EEF in the direction of turning. This motion is much simpler, and an operator can perform it much more quickly and easily. Figure 5.1 shows an example of the input space reduction enabled by the addition of compliance parameters to Affordance Templates.

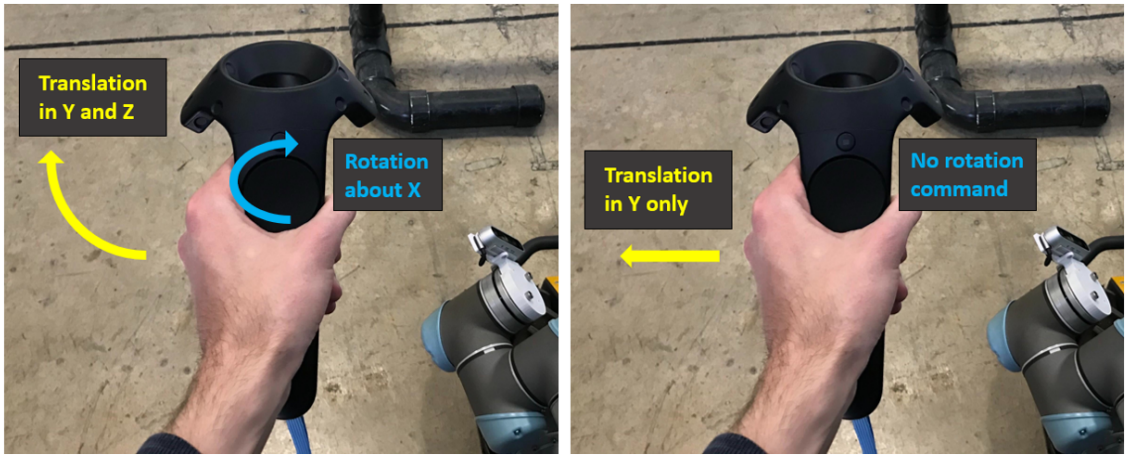


Figure 5.1: Left, without compliance the user must trace the arc, shown in yellow, whist also controlling the rate of rotation, shown in blue, to close the valve. Right, with compliance enabled in the roll dimension the user needs only move the controller laterally, shown in yellow, and does not need to worry abouot rotation when closing the valve.

Once an Affordance Primitive has been created for one task, it should be easy to modify the compliance parameters to work for all tasks of the same type, allowing for differences in the rigidity of the environment. Ideally, the system would be used exclusively in one facility and parameter archiving could be incorporated into the primitives at specific locations within the facility. For example, if the robot was being used in a facility with 6 valves of the same size and type, it would be useful to store information for each valve in case some are easier to turn than others. The baseline Affordance Primitive would remain the same, but the robot would use the location data to modify the compliance parameters according to its memory of previously completing the task at each location. This customization further automates the process of cleanly performing tasks around a large facility, and further reduces the level of human supervision necessary to perform contact tasks.

With industries moving towards digitized facilities, the implementation of Affordance Primitives with associated parameter archiving make more sense than ever. Plants have begun implementing a network of sensors across most pieces of equipment, which can be used to update information about equipment in real time. If the sensor data is available to the robot at the time the template is to be placed, the robot can use that information to automatically register the template to the equipment. Similarly, any abnormalities noted while turning the valve could be stored in the facility records associated with that piece of equipment and used in future executions of the task.

Eventually, as ATs become more advanced, the stored data could be used to automatically modify the template based on the current state. For example, if the

valve was neither fully open nor closed and the user wanted the valve to be turned to another position that was neither fully open nor closed, the sensor data could be used to adapt the AT to achieve the new goal. For a simple task like turning a ball valve, this would be relatively simple as the valve has a limited range of motion. Knowing the physical characteristics of the valve along with the starting and desired final positions of the lever, it is simple to mathematically calculate the proper waypoint locations to achieve the turn. This opens the possibility of performing a much greater range of tasks than is currently possible.

5.2 Parameters

Section 5.1 explains that Affordance Primitives are Affordance Templates with integrated compliance parameters, but it does not explain what those parameters are, nor their importance. This section breaks down the structure of an AP and explains each parameter's purpose.

Affordance Primitives' compliance parameters are structured as tuples, which are finite ordered lists. As described in [30], the AP compliance tuples consisting of consisting of eleven parameters:

- $k \in \mathbb{R}^m$: The impedance "stiffness" relating applied wrench to EEF velocity and forming the diagonal matrix K from (2.6)
- $\beta \in \mathbb{R}^m$: The impedance "damping" relating the time derivative of applied wrench to EEF velocity and forming the diagonal matrix B from (2.6)
- $\xi_{apply} \in \mathbb{R}^m$: The desired applied wrench in (2.6), given in N and $N \cdot m$

- $\xi_{max} \in \mathbb{R}^m$: The maximum allowable force/torque in each direction in N and $N \cdot m$
- $F_{max} \in \mathbb{R}^1$: The maximum combined force in N
- $\tau_{max} \in \mathbb{R}^1$: The maximum combined torque in $N \cdot m$
- $\Delta x_{max} \in \mathbb{R}^m$: The maximum allowable displacement from compliance
- $\dot{x}_{max} \in \mathbb{R}^m$: The maximum allowable EEF velocity
- $\dot{q}_{max} \in \mathbb{R}^n$: The maximum allowable joint velocities in (2.7)
- $\nu \in_2^m$: The jogging control dimensions allowed in (2.7)
- $\mu \in_2^m$: The compliant dimensions in (2.7)

Where ν reduces the input space and μ reduces the compliance space.

The short-term goal of APs is not to modify ATs, but rather to add functionality on top of the existing AT structure. This added layer does not currently change the way ATs work in terms of planning between waypoints, but in the long-term ATs will need to be restructured if Affordance Primitives are going to be utilized to their fullest potential.

5.3 Demonstration

To show the potential of our APs, we conducted a study of novice users turning a single ball valve. This study was intended as a proof of concept that automated compliance and input reduction would greatly reduce the difficulty of a complex task.

For the purposes of this demonstration, the valve turn task was decomposed into two sub-tasks: the grasp and the turn. Users were tasked with turning the valve as quickly as possible while attempting to avoid safety faulting the arm. If a safety fault did occur (which happened for every user at least once), the user did not have to restart the task, but they did lose time because the arm had to be reset.

For this study, we had each user perform the valve turn with different levels of compliance active. For the first iteration of the test, the operators were working with compliance fully disabled. The second trial had compliance enabled, but there was no reduction in the input space. Finally, the third trial was performed with compliance enabled and with input space reduction. The compliance parameters ν are shown in Table 5.1. Table 5.2 shows which compliance parameters and directions of motion μ were enabled for the grasping sub-task and Table 5.3 shows the same information for the turning sub-task.

At this point in time, we did not have access to the Affordance Template software packages discussed in 3, so the participants were performing the task via jogging. For jogging, users were given the choice between using a SpaceNav 3D controller and/or VR hand controllers, with the option to switch between the two devices at any time during their run. The controls for these devices are shown in Figure 4.5 and Figure 4.4. As the users for this study were all novice, they were given a few minutes before the first trial to become comfortable with the jogging controls after the task was explained to them.

To measure the success of the task, candidates were evaluated on several metrics: task completion, time to complete, total number of safety faults during task

Table 5.1: AP Parameters Shared with Grasp and Turn

Parameter	**X**	**Y**	**Z**	**Roll**	**Pitch**	**Yaw**
k	8000	1000	1000	5	40	60
β	50000	10000	10000	300	600	600
ξ_{apply}	0	0	0	0	0	0
ξ_{max}	80	80	80	60	60	60
Δx_{max}	0.15	0.15	0.05	$\pi/2$	$\pi/16$	$\pi/16$

Table 5.2: AP Parameters for Valve Grasp

Parameter	**X**	**Y**	**Z**	**Roll**	**Pitch**	**Yaw**
ν	1	1	1	1	1	1
μ	1	1	0	1	0	0

performance, and user opinion of difficulty (measured on a scale of 1 to 5 with 1 being the easiest and 5 being the hardest). We wanted primarily quantitative measures to evaluate the degree to which our APs aided users, but we also wanted a qualitative measure of perceived difficulty. The results of this user study are shown in Table 5.4 and Figure 5.2.

For the first iteration of the task – the "no compliance" iteration – most users were able to complete the task, but one user gave up without finishing due to frustration. All users incurred multiple safety faults regardless of completion status. The average time to complete the task (not counting any incomplete attempts) was 168.6 seconds, the average number of safety faults was 4.9, and the average user opinion of difficulty was 4/5.

During the second trial compliance was enabled, but there was not any re-

Table 5.3: AP Parameters for Valve Turn

Parameter	**X**	**Y**	**Z**	**Roll**	**Pitch**	**Yaw**
ν	0	1	0	1	0	0
μ	1	0	0	1	0	0

duction in the input space. This reduced the task difficulty by controlling the EEF torques and forces, but it did not reduce the task to its simplest set of input commands. With compliance added, there was still one user that failed to complete the task (the same user from the first iteration), but the average completion time, number of safety faults, and user opinion of difficulty were all reduced. In this iteration, the average completion time fell to 135.6 seconds, the number of safety faults fell to 3.0 per user, and the perceived difficulty fell to 3.25/5.

The final iteration of the test had compliance fully enabled along with selective motion to reduce the task input space. With the Affordance Primitive fully enabled, all users were able to successfully complete the valve-turn task. Additionally, the average task completion time was further reduced over the compliance-only trial down to 81.8 seconds, the safety faults were reduced to 1.4 faults per person, and the perceived difficulty was reduced to 2.25/5.

Table 5.4: Testing Result Averages

Trial	**Time (s)**	**Safety Faults**	**Difficulty (1-5)**
Trial 1	168.6	4.9	4
Trial 2	135.6	3.0	3.25
Trial 3	81.8	1.4	2.25

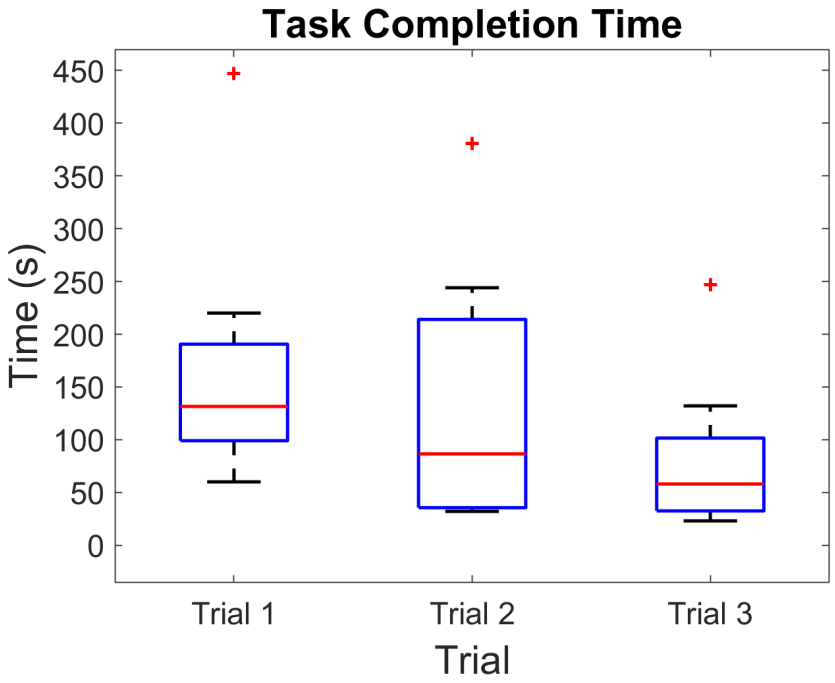


Figure 5.2: Utilizing Affordance Primitives reduced the time to complete and difficulty of the valve turn task (n=8).

Figure 5.2 and Table 5.4 summarize the results of the user study. The addition of the compliance and directional control cut the completion time for the valve turn in half and reduced safety faults to one-third compared to the trial without compliance.

As shown in Table 5.2 and Table 5.3, the grasp and turn sub-tasks had different optimal compliance parameters. The compliance parameters for the grasp sub-task were enabled for the X, Y, and Roll dimensions, which accounted for most minor misalignments, as shown in Figure 5.3. If the gripper was slightly to the left or the right of the valve before closing, the compliance in the Y dimension would drag the gripper into place when closed. If the gripper was slightly rotated, the roll compliance auto-

matically caused the gripper to right itself upon closing. In both cases, misalignments like these would normally cause safety faults during the grasp if the user did not take the time to manually correct them. For the turning sub-task, compliance was only enabled for the roll and X dimensions. Compliance in the X dimension prevented the gripper from exerting large forces by pulling or pushing on the valve during the turn, which is easy to do if the gripper is not perfectly orthogonal to the valve. The roll compliance, in combination with the directional motion being limited to the Y dimension (tangent to the valve axis in the direction of turning), made it such that the user only needed to control movement in the Y direction and the arm would follow the tangent arc with the gripper freely rotating as necessary. The reduced input space described above is shown in Figure 5.4.

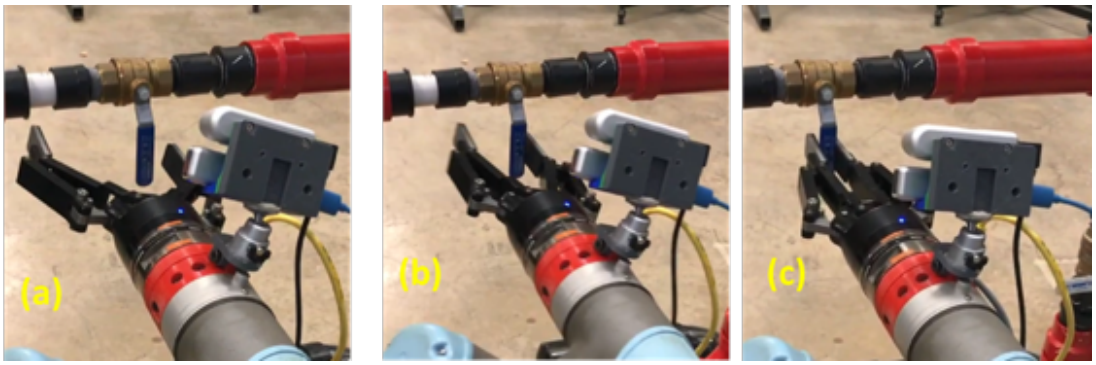


Figure 5.3: In (a), the user has located gripper around the valve with significant offset and rotational errors. In (b), the gripper has started to close and makes contact with the valve with one finger before (c) the affordance compliance parameters assure the controller continues to grasp the handle while compliantly correcting the operator's positioning errors [30].

By incorporating these compliance parameters into the template during its creation, a novice user will be able to execute the Affordance Template with compli-

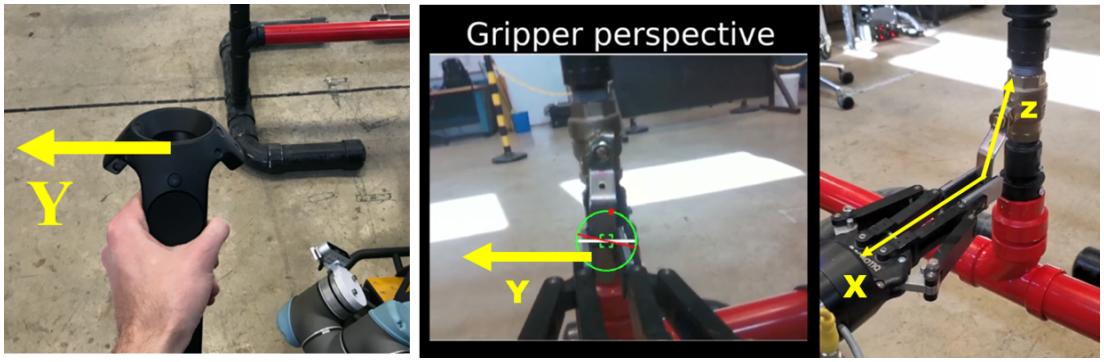


Figure 5.4: Left, the user simply moves (a swipe gesture) the controller to the left, while (middle, right) the developed controller utilizes compliance to assure the gripper correctly tracks the rotation and elevation in the grasp point as the valve is closed [30].

ance active and optimized for each subset of the task. This serves to further reduce the barrier to entry for non-experts to be able to utilize robots in their daily jobs. The potential for any employee, regardless of technical experience, to be able to supervise a robot performing a task makes it a much more attractive solution for industries to pursue.

5.4 Applications/Future Work

One future application of Affordance Primitives is to automatically generate Affordance Templates by "recording" the execution of an example task. More plainly, these primitives will allow the operator to populate the compliance parameters of an AP by manually performing the task. Recording new templates would be a far simpler process than the current method of adding waypoints manually (using physical measurements) or by adding the waypoints using the robot pose, which is more efficient, but still requires the user to jog or plan to the waypoint locations and then add

the pose data. By recording and saving a sample run-through of the task, operators would have a record of the EEF pose at every point during the task as well as a list of the jog commands required for successful execution. This data would then be converted into a complete Affordance Template for future use.

In nearly all cases, this recording capability would allow the operator to set up the AP comfortably in a controlled environment like a research lab and then be able to apply the template in the field. Speaking from firsthand experience, allowing the operator to create the template in a local environment considerably diminishes the time required to set up a template. Performing the task locally is much easier as the operator does not need to rely on sensor data and situational awareness aids and can simply jog the arm to the proper waypoint positions and use the current robot pose to populate the template. Additionally, some time is saved because less time is needed to reset the manipulation object between testing attempts.

Another application for APs is any task where the trajectory taken by the arm needs to be extremely precise and consistent. One weakness of ATs is that they rely on motion planners such as MoveIt to plan between their waypoints. As mentioned earlier, MoveIt does not give the user control over the path between points and is not guaranteed to find the best path or the same path every time when planning between the same two points. If the contact task in question requires the manipulator to plan through a tight space, the only current option is to add a large number of waypoints so that the motion from each waypoint to the next is very small and unlikely to generate a poor plan. Even so, it is impossible to guarantee that the planner never violates the constraints of the task. Additionally, adding too many waypoints is impractical

both computationally and time-wise as a large number of waypoints would waste a significant amount of time over other methods where the user does control the trajectory, such as jogging. A possible solution to this issue is to modify the AT packages to work using a series of jog commands as opposed to a series of waypoints. While this would undoubtedly be complicated to implement, it would allow much finer control over the trajectory of the EEF during task execution.

In their current form, Affordance Templates, due to their waypoint nature, are not capable of utilizing Affordance Primitives to their fullest potential. Specifically, the waypoint restriction of AT packages limits the effectiveness of the input reduction capabilities provided by APs. Going back to the valve example mentioned earlier, the AP input reduction allows the user to complete the valve turn simply by jogging in one direction. This input command does not work with AT packages, which must be adapted moving forward to allow for the use of jogging commands as an alternative option to waypoints for motion planning. Additionally, altering AT packages to allow for jogging will allow for easier population of ATs through the recording of jogged motion.

ATs are open-loop solutions, which limits their usefulness for tasks that require feedback, i.e. pressing a gas pedal to reach and maintain a certain speed or turning a valve to reach and maintain a certain pressure of the fluid inside. With the current waypoint implementation tasks like this would be forever unattainable. If the templates were reworked to use jog commands, templates would be able to close the control loop with equipment sensors and perform a wider variety of contact tasks.

5.5 Summary

Affordance Primitives integrate compliance parameters into their template structure to reduce the rate of template failure due to misalignment. These parameters can be set independently at each waypoint so that each motion can be optimized with respect to compliance. This compliance optimization allows industries to be more confident that the robot can safely and successfully perform tasks with minimal supervision.

Additionally, APs aim to expand the number of tasks that they can be used to perform. Currently, templates can only be used for open-loop tasks. Any task that requires regulating something via a feedback loop is impossible with existing AT implementations. For example, if you wanted to reach a desired pressure in a pipe by opening or closing a valve it would be impossible with the current waypoint structure. To make a wider variety of tasks achievable, ATs will need to implement jogging.

Finally, future APs will be created by recording the successful execution of a task using the jogger. Current affordance templates allow for the user to pull waypoint coordinates from the current pose of the robot, but recording jog commands would make affordance templates much smoother and more reliable. As mentioned previously, MoveIt is currently used to plan between template waypoints, but it is not guaranteed to give the same plan every time (sometimes plans are produced that violate joint limits). By recording the exact jog commands necessary to perform the task, it is guaranteed that the template will execute the same trajectory every time.

Chapter 6

Conclusions

6.1 Research Summary

Contact tasks are the next critical capability remote robotic systems must be able to reliably and easily complete in industrial environments. The ability to have robots perform manipulation tasks remotely will greatly improve employee safety in hazardous industries, from Nuclear to Oil & Gas. In addition to enhanced safety, improved autonomy will allow human operators to supervise multiple tasks/processes at the same time, improving overall efficiency over the current method of teleoperation.

The difficulty of contact tasks lies in the need to closely monitor and manage contact forces at the robot's end effector, which is difficult without user experience, haptic feedback, or manipulator compliance. User experience is often lacking as many industrial facilities do not employ trained roboticists full time, meaning that robot agents would only be used on occasion which would not justify their expense. Haptic feedback requires significant fiscal investment to implement for all robots on a facility, requires constant monitoring each time a task is completed, and still requires some experience to accurately interpret. Thus, a compliant controller was implemented to automatically manage end effector contact forces and make complex tasks less burdensome for operators, enabling task completion by an employee with minimal

training. With the compliant controller running, the user is prevented from exceeding the safe force limit set for the manipulator, reducing the chance of damage to the manipulator, manipulated object, or the environment.

Ideally, tasks would be done purely autonomously with a remote human operator supervising several agents at once. Fully-autonomous manipulation agents would require the development of high-level mission planning algorithms specific to the needs of each industry and company. The first step to safely achieving full autonomy in hazardous environments is to automate task planning, which removes significant burden from the operator, but still grants them some control over the tasks performed.

In this research, a wide variety of action planners were evaluated, and ATs were determined to be the best option for implementing pseudo-autonomous behavior with novice users. With their focus on intuitive graphical user interfaces and the capability to complete multi-step tasks, Affordance Templates present the best option for employee-supervised semi-autonomous task completion.

Affordance Templates are 3D visualizations of task objects that contain EEF waypoints to perform pre-defined task actions. They are typically created in advance by experienced users, but they are infinitely re-usable and can be created and operated by someone of any skill level. This is ideal for an industry such as the Oil & Gas industry, where a limited number of employees work at production facilities and none are trained robotics experts. Thus, an easy-to-use software package such as ATs is pivotal in robots' being used for routine, albeit dangerous tasks.

Overall, ATs provide a good balance between autonomy and user supervision in

hazardous environments, especially where delicate or sensitive equipment is involved. Additionally, they do not require advanced knowledge of an environment to be used. Operators can use existing templates, or create their own templates relatively quickly to address new or evolving challenges.

Two Affordance Template software packages, CRAFTSMAN and UseIt, were implemented and evaluated for use in hazardous environments. These packages both allow for the rapid generation of templates to complete complex tasks. CRAFTSMAN is the more mature product, and supports more advanced capabilities, but is more complicated in its use. UseIt, conversely, is exceptionally easy to use, with little training required to run or generate templates, but has more limited capabilities. Therefore, the author recommends CRAFTSMAN be used by more experienced roboticists for more complex tasks, and UseIt be used by everyday employees for more routine tasks[1].

A significant issue with ATs is their low tolerance for misalignment, which can be caused by user error or sensor bias. In conditions with low lighting or poor data transmission, it is imperative that templates be as robust to misalignments as possible. To achieve this, a compliance controller was implemented with ATs to allow for more flexibility during contact with objects. This controller allowed operators to both control the dimensions of compliance, optimizing task performance and decreasing the likelihood of operator error resulting in the EEF impacting the environment, and set the maximum EEF force, further minimizing the risk to the robot and facility

[1]UseIt 2.0, which is currently under development, will allow for the generation of more complex templates. Thus, it will be more similar to CRAFTSMAN in terms of its sophistication.

hardware.

To combat this misalignment issue and increase the placement tolerance for templates, a compliant controller was integrated with ATs. This took place in the form of a remote demonstration from Austin, Texas to Perth, Western Australia. The operator tested the affects of ATs vs manual control and compliance vs no compliance. In the end, ATs proved to be faster and less error prone than manual control, and enabling compliance further reduced errors and the set-up time for tasks as their added flexibility helped forgive most minor misalignments.

At this point in time, ATs suffer from a scope problem. There is no standard for how large or how limited the scope of a template should be. In different scenarios, one might prefer a template with a very wide scope if it is the only task of its type at that facility. For example, if a small facility contained eighteen ball valves, but only one double block and bleed setup (three of the ball valves), it would make sense to have two templates. First, they would want a minimalist template for the generic ball valves since there are so many, but since there is only one double block and bleed apparatus as opposed to several, they would probably also want a template encompassing that task as a whole. By making a large template for the DBB, it simplifies the task of the user, who only has to align the single template as opposed to needing to align three and order them appropriately in terms of chronological execution.

The scope problem is exacerbated by one of the largest limitations of ATs, which is that they allow only one template to be loaded into the environment at a time. This limitation lends itself to large, unrealistically-specific templates as opposed

to the more generalized, minimalist templates that are employable in a wide range of environments. In the future, it is the author's opinion that a priority should be placed on allowing for multiple object models to be added to the virtual environment, with some mechanism implemented to "schedule" the templates to complete in a user-specified order. This would facilitate larger tasks being represented as "macro templates" as opposed to a single, overly-complex template. Current AT development is working to address this limitation.

Finally, Affordance Primitives were introduced, which integrate compliance parameters directly into Affordance Templates. This integration makes rigid manipulators flexible, allowing for some misalignment when using an AT to perform a task, and thus reduce task set up time. Additionally, Affordance Primitives' compliance can reduce the input space of a task down to simpler motions. With compliance parameters optimized, a valve turn that would normally require an arc motion combined with a synchronized rate of rotation could be simplified to single lateral input command. This reduced input space also lends itself well to jogging commands as opposed to point-to-point commands that are currently used to plan between waypoints.

6.2 Future Work

As seen in Section 3, both options evaluated have some advantages over the other and thus document incremental improvements that could occur in each package. In addition, standard improvements related to improved interfaces, increased hardware agnosticism, improved documentation, computational efficiency, etc. are possible in both cases. There are a also few unexplored avenues for future research

and development.

One major area for potential growth is the automatic registration of affordances to environment objects. In the implementation described in Chapter 4, we manually aligned the template using an AR marker. While the AR marker makes the alignment task slightly easier, it would be far better to automate the process using computer vision combined with localization of the robot on the facility. Beyond just recognizing and registering a template to the pre-selected object, it would be very useful for the robot to be able to analyze objects and discern the available affordances for the object based on its shape. While this would be less useful in an industrial setting, where each piece of equipment is cataloged and its uses known, it would be very useful when operating in new or unknown environments. A particularly relevant example is the robots inspecting the Fukushima site, where items were well-modeled but catastrophically damaged.

Conversely, for human-in-the-loop control, another potential area of future development is to integrate ATs in VR. If an operator was creating a new template on the fly for a remote system, it would be very useful to have VR for increased situational awareness. To be able to align the template model with its real-world counterpart, the robot must already be equipped a 3D sensor such as a LiDAR or a depth camera, which returns a 3D point cloud. This point cloud could be displayed in the VR headset worn by the operator, which would allow that operator to more easily align the template. A VR headset for template alignment would be most valuable to industrial employees with limited computer experience.

Additionally, ATs would benefit from the option of using a string of jog com-

mands instead of planning between waypoints. As mentioned in Chapter 5, a future application of Affordance Primitives is to record the successful execution of a task and be able to save it as a new template or primitive. The integration of jog commands would make this template/primitive recording process even simpler, as the user would not need to halt execution mid-task to specify desired waypoints. Using jog commands would also ensure the template's repeatability as the jog commands would be guaranteed to produce identical end effector motion each time the template is executed, as opposed to point-to-point planning, which can generate several plans between a set of two points, not all of which are ideal. Template scaling would potentially be more complicated using jog commands, but not prohibitively so and the benefits of jog commands greatly outweigh this potential downside.

Another avenue that AT development can explore is "mission planning". This would require the ability to load multiple templates into an environment and schedule them in some manner. This schedule could be rigid - perform tasks A, B, C, and D in that order - or could be more flexible - perform task A before task D, but other tasks can be performed in any order. Stringing multiple ATs together to achieve an extended task or set of tasks would be a step closer to full autonomy. The mission plan would be set up at the beginning of the day - or if the tasks are performed daily or weekly - set up to repeat on a cycle. Mission planning would primarily be used for routine maintenance or surveillance tasks, and setting up mission plans would eliminate the need for operators to send commands to the robot multiple times per day. The more tasks the robot is responsible for performing, the more benefit there is to mission planning. This would also require the AT template software to support

navigation.

To make mission planning even more comprehensive, ATs could be further expanded to work with multiple robots or autonomous agents. This would require running a multimaster system in ROS and only sending required messages between each robot and the AT software. The AT packages discussed in this work currently only support single or dual-arm planning, so planning with multiple agents would require a significant increase in their complexity. Expanding ATs to employ multiple robots would significantly increase both the quantity and complexity of tasks that can be automated. For example, multiple agents could be deployed to perform emissions testing in multiple locations simultaneously. Additionally, with multiple agents available the operator could choose a particular agent to perform a task, or choose a particular type of agent and allow factors such as distance from the task and battery life to be the deciding factors as to which agent is chosen.

Finally, ATs need to close the loop in terms of their control framework. Their current open-loop implementation prevents them from performing any task that relies on sensor feedback. Enabling a closed-loop structure would allow ATs to perform tasks like maintaining a certain speed while driving a car uphill or downhill by regulating the force being applied to the gas pedal. In the context of industrial automation, this closed-loop framework could be used to maintain the pressure of a process fluid in a pipe by opening or closing a valve. With industries blanketing their facilities in sensor suites that provide real-time data, performing this type of task is only limited by ATs open-loop structure.

6.3 Final Remarks

This work successfully completed all the objectives outlined in Chapter 1. In Chapter 3, the Affordance Template packages CRAFTSMAN and UseIt were installed and tested in simulation to evaluate their capabilities and ease of use. Additionally, the compliant controller detailed in Chapter 2 was integrated and tested with UseIt. This demonstration of UseIt with the compliant controller, discussed in Chapter 4, confirmed that compliance reduced the difficulty of performing contact tasks with Affordance Templates. Finally, Chapter 5 introduced our concept of Affordance Primitives and described their advantages over current AT packages.

While there are still many advancements to be made in the automation of remote contact tasks, this work provides a clear path to increased autonomy for remote systems in industrial environments that can be implemented in the short term.

Appendix A

Lessons Learned

A.1 Advice for Implementation/Integration of New Software

This project required a great deal of setting up software that was previously only accessible to NASA and TRACLabs developers. This meant that I was a "guinea pig" when it came to installing and building these software packages. Luckily, both teams provided concise installation instructions tailored to my system, which reduced the amount of configuration files than I needed to edit.

CRAFTSMAN's packages can be downloaded as ROS debians, which the user needs to be granted special access to download. Once that access has been granted, installation is simple. The user uses the "apt-get install" command in the same manner that all other ROS debians are installed. After installation, CRAFTSMAN provides a useful set up wizard to help users make the necessary configuration file edits to get their robot running with CRAFTSMAN. The process is very simple, with little room for user error.

UseIt's ROS packages are all stored on GitHub. Once granted access to the proper NASA repository, the user can download the vcs text file, which lists the necessary packages, the correct branch, and the link to the package. To install all of the packages in one step, the user needs to "vcstool" and run "vcs import vcs.txt",

which downloads all of the ROS packages on the correct branch. After installation, the user has to make adjustments to the configuration files for the robot and the gripper to match their hardware. UseIt's installation process is slightly more involved than that of CRAFTSMAN, but is still relatively simple.

When installing software packages such as these that require different pip and python versions, it is advisable to start with two clean Ubuntu installs. This could be on two separate computers, on different partitions on the same computer, or on virtual machines. CRAFTSMAN and UseIt run different versions of pip, making them incompatible on the same environment and meaning that fixing one build breaks the other. Luckily, this was not a setback in this project, but it could cause issues for someone who installs both on the same machine not knowing they are incompatible.

When installing and using any new software, keep a detailed log of daily progress. Write out all installation commands, package versions, and any changes made to the code or launch files. When something works properly take screenshots or screen recordings to help you remember several months down the line and to document it for future users. When using a screen recorder, make sure the pointer is being captured and sound is being recorded as well. Sound can be edited out later if necessary and is very useful to have access to when reviewing your old work. When it comes to teaching or leaving documentation for the next person to learn how to use your software, a list of instructions is great, but a video showing the process is even easier to follow. Finally, keep track of any software updates you perform as they may lead to something breaking, and you'll want to know what changed so you can try to revert the changes.

Bibliography

[1] Defense Advanced Research Projects Agency. Darpa robotics challenge (drc). https://www.darpa.mil/program/darpa-robotics-challenge. Accessed: 2020-09-22.

[2] Patrick Beeson and Barrett Ames. Trac-ik: An open-source library for improved solving of generic inverse kinematics. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 928–935. IEEE, 2015.

[3] Denis Borenstein. A directed acyclic graph representation of routing manufacturing flexibility. *European journal of operational research*, 127(1):78–93, 2000.

[4] ChuXin Chen and Mohan M Trivedi. Task planning and action coordination in integrated sensor-based robots. *IEEE transactions on systems, man, and cybernetics*, 25(4):569–591, 1995.

[5] Luiz SH De Mello and Arthur C Sanderson. And/or graph representation of assembly plans. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 1986.

[6] Maurice Fallon, Scott Kuindersma, Sisir Karumanchi, Matthew Antone, Toby Schneider, Hongkai Dai, Claudia Pérez D'Arpino, Robin Deits, Matt DiCicco, Dehann Fourie, et al. An architecture for online affordance-based perception and whole-body planning. *Journal of Field Robotics*, 32(2):229–254, 2015.

[7] Richard E Fikes and Nils J Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208, 1971.

[8] Terrence Fong, Charles Thorpe, and Charles Baur. *Collaborative control: A robot-centric model for vehicle teleoperation*, volume 1. Carnegie Mellon University, The Robotics Institute Pittsburgh, 2001.

[9] Lucas Eddie Gallegos III. *A demonstration and comparative analysis of haptic performance using a Gough-Stewart platform as a wearable haptic feedback device.* PhD thesis, 2019.

[10] Michael Gelfond and Vladimir Lifschitz. Representing action and change by logic programs. *The Journal of Logic Programming*, 17(2-4):301–321, 1993.

[11] James J Gibson. The theory of affordances. *Hilldale, USA*, 1(2), 1977.

[12] David Gossow, Adam Leeper, Dave Hershberger, and Matei Ciocarlie. Interactive markers: 3-d user interfaces for ros applications [ros topics]. *IEEE Robotics & Automation Magazine*, 18(4):14–15, 2011.

[13] Stephen Hart, Paul Dinh, and Kimberly Hambuchen. The affordance template ros package for robot task programming. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 6227–6234. IEEE, 2015.

[14] Stephen Hart, Paul Dinh, and Kimberly A Hambuchen. Affordance templates for shared robot control. In *2014 AAAI Fall Symposium Series*, 2014.

[15] HEBI. X-series acuator. https://www.hebirobotics.com/x-series-smart-actuators. Accessed: 2020-06-11.

[16] Tucker Hermans, James M Rehg, and Aaron Bobick. Affordance prediction via learned object attributes. In *IEEE International Conference on Robotics and Automation (ICRA): Workshop on Semantic Perception, Mapping, and Exploration*, pages 181–184. Citeseer, 2011.

[17] Neville Hogan. Impedance control: An approach to manipulation. In *1984 American control conference*, pages 304–313. IEEE, 1984.

[18] Joshua James, Yifan Weng, Stephen Hart, Patrick Beeson, and Robert Burridge. Prophetic goal-space planning for human-in-the-loop mobile manipulation. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 1185–1192. IEEE, 2015.

[19] Matthew Johnson, Jeffrey M Bradshaw, Paul J Feltovich, Catholijn M Jonker, M Birna Van Riemsdijk, and Maarten Sierhuis. Coactive design: Designing support for interdependence in joint activity. *Journal of Human-Robot Interaction*, 3(1):43–69, 2014.

[20] Twan Koolen, Jesper Smith, Gray Thomas, Sylvain Bertrand, John Carff, Nathan Mertins, Douglas Stephen, Peter Abeles, Johannes Englsberger, Stephen Mccrory, et al. Summary of team ihmc's virtual robotics challenge entry. In *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 307–314. IEEE, 2013.

[21] Dirk Kraft, Norbert Krüger, Florentin Wörgötter, Christopher Geib, Ron Petrick, Mark Steedman, Renaud Detry, Justus Piater, Ales Ude, Tamim Asfour, et al. Deliverable no.: D8. 1.5 title of the deliverable: Publication about multi-level learning sys-tem.

[22] Norbert Krüger, Christopher Geib, Justus Piater, Ronald Petrick, Mark Steedman, Florentin Wörgötter, Aleš Ude, Tamim Asfour, Dirk Kraft, Damir Omrčen, et al. Object–action complexes: Grounded abstractions of sensory–motor processes. *Robotics and Autonomous Systems*, 59(10):740–757, 2011.

[23] Joohyung Lee, Vladimir Lifschitz, and Fangkai Yang. Action language bc: Preliminary report. In *IJCAI*, pages 983–989. Citeseer, 2013.

[24] Frank L Lewis, Darren M Dawson, and Chaouki T Abdallah. *Robot manipulator control: theory and practice*. CRC Press, 2003.

[25] Jorge Lobo, Gisela Mendez, and Stuart R Taylor. Knowledge and the action description language a. *arXiv preprint cs/0404051*, 2004.

[26] Pat Marion, Maurice Fallon, Robin Deits, Andrés Valenzuela, Claudia Pérez D'Arpino, Greg Izatt, Lucas Manuelli, Matt Antone, Hongkai Dai, Twan Koolen, et al. Director: A user interface designed for robot operation with shared autonomy. *Journal of Field Robotics*, 34(2):262–280, 2017.

[27] Troy McMahon, Odest Chadwicke Jenkins, and Nancy Amato. Affordance wayfields for task and motion planning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2955–2962. IEEE, 2018.

[28] MoveIt. Moveit motion planning framework. https://moveit.ros.org/. Accessed: 2020-07-22.

[29] Shimon Y Nof. *Handbook of industrial robotics*. John Wiley & Sons, 1999.

[30] Adam Pettinger, Cassidy Elliott, Pete Fan, and Mitch Pryor. Reducing the teleoperator's cognitive burden for complex contact tasks using affordance primitives. In *Proceedings of the IEEE International Conference on Robots and Systems*. IEEE, 2020.

[31] J Rocha and C Ramos. Plan representation and generation for manufacturing tasks. In *Proceedings. IEEE International Symposium on Assembly and Task Planning*, pages 22–27. IEEE, 1995.

[32] Joao Rocha and Carlos Ramos. Representing and generating operation sequences for manufacturing tasks. In *Proceedings of the 1997 IEEE International Symposium on Assembly and Task Planning (ISATP'97)-Towards Flexible and Agile Assembly and Manufacturing-*, pages 134–139. IEEE, 1997.

[33] Alberto Romay, Stefan Kohlbrecher, David C Conner, Alexander Stumpf, and Oskar von Stryk. Template-based manipulation in unstructured environments for supervised semi-autonomous humanoid robots. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 979–986. IEEE, 2014.

[34] Erol Şahin, Maya Çakmak, Mehmet R Doğar, Emre Uğur, and Göktürk Üçoluk. To afford or not to afford: A new formalization of affordances toward affordance-based robot control. *Adaptive Behavior*, 15(4):447–472, 2007.

[35] Mark W Spong, Seth Hutchinson, Mathukumalli Vidyasagar, et al. *Robot modeling and control*. 2006.

[36] Leila Takayama, Wendy Ju, and Clifford Nass. Beyond dirty, dangerous and dull: what everyday people think robots should do. In *2008 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 25–32. IEEE, 2008.

[37] TRACLabs. Craftsman. https://traclabs.com/projects/craftsman/. Accessed: 2020-09-14.

[38] TRACLabs. Craftsman documentation. https://traclabs.bitbucket.io/. Accessed: 2020-09-15.

[39] TRACLabs. trac_ik documentation. https://bitbucket.org/traclabs/trac_ik/src/master/. Accessed: 2020-010-11.

[40] Robert Valner, Karl Kruusamäe, and Mitch Pryor. Temoto: intuitive multi-range telerobotic system with natural gestural and verbal instruction interface. *Robotics*, 7(1):9, 2018.

[41] Robert Valner, Veiko Vunder, Andy Zelenak, Mitch Pryor, Alvo Aabloo, and Karl Kruusamäe. Intuitive "human-on-the-loop" interface for tele-operating remote mobile manipulator robots, 2018.

[42] Rusty Alexander von Sternberg. GCCF: A Generalized Contact Control Framework. Master's thesis, The University of Texas at Austin, 2016.

[43] Joshua Murry Williams. Automated conceptual design of manufacturing work-cells in radioactive environments. 2013.

[44] Natsuki Yamanobe, Weiwei Wan, Ixchel G Ramirez-Alpizar, Damien Petit, Tokuo Tsuji, Shuichi Akizuki, Manabu Hashimoto, Kazuyuki Nagata, and Kensuke Harada. A brief review of affordance in robotic manipulation research. *Advanced Robotics*, 31(19-20):1086–1101, 2017.

[45] Andy Zelenak, Robert G Reid, Adam Pettinger, and Mitch Pryor. Reactive motion control for real-time teleoperation and semi-autonomous contact tasks.

Vita

Cassidy Morgan Elliott was born in Huntsville, Alabama. She graduated summa cum laude with her Bachelor's degree in mechanical engineering from the University of Alabama in 2018. In the fall of that year she joined the Nuclear and Applied Robotics Group at the University of Texas at Austin and was awarded the Provost's Excellence Fellowship and the Thrust 2000 - H. R. Crawford Endowed Graduate Fellowship in Engineering.

Email address: cmelliott2@utexas.edu

This thesis was typeset with LaTeX[†] by Cassidy Morgan Elliott.

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