Autonomous Task-Based Grasping for Mobile Manipulators

by

Michael Hegedus

MASc, University of Regina, 2009

BASc, University of Regina, 2006

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
School of Engineering Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY

Spring 2021

Declaration of Committee

Name:	**Michael Hegedus**
Degree:	**Doctor of Philosophy**
Title:	**Autonomous Task-Based Grasping for Mobile Manipulators**
Committee:	**Chair:** Michael Sjoerdsma Senior Lecturer, Engineering Science
	Kamal Gupta Co-Supervisor Professor, Engineering Science
	Mehran Mehrandezh Co-Supervisor Professor, Industrial Systems Engineering University of Regina
	Carlo Menon Committee Member Professor, Engineering Science
	Mirza Faisal Beg Examiner Professor, Engineering Science
	Hong Zhang External Examiner Professor, Computing Science University of Alberta

Abstract

A fully integrated grasping system for a mobile manipulator to grasp an unknown object of interest (OI) in an unknown environment is presented. The system autonomously scans its environment, models the OI, plans and executes a grasp, while taking into account base pose uncertainty and obstacles in its way to reach the object. Due to inherent line of sight limitations in sensing, a single scan of the OI often does not reveal enough information to complete grasp analysis; as a result, our system autonomously builds a model of an object via multiple scans from different locations until a grasp can be performed. A volumetric next-best-view (NBV) algorithm is used to model an arbitrary object and terminates modelling when grasp poses are discovered on a partially observed object. Two key sets of experiments are presented: i) modelling and registration error in the OI point cloud model is reduced by selecting viewpoints with more scan overlap, and ii) model construction and grasps are successfully achieved while experiencing base pose uncertainty.

A generalized algorithm is presented to discover grasp pose solutions for multiple grasp types for a multi-fingered mechanical gripper using sensed point clouds. The algorithm introduces two key ideas: 1) a histogram of finger contact normals is used to represent a grasp “shape” to guide a gripper orientation search in a histogram of object(s) surface normals, and 2) voxel grid representations of gripper and object(s) are cross-correlated to match finger contact points, i.e. grasp “size”, to discover a grasp pose. Constraints, such as collisions with neighbouring objects, are incorporated in the cross-correlation computation. Simulations and preliminary experiments show that 1) grasp poses for three grasp types are found in near real-time, 2) grasp pose solutions are consistent with respect to voxel resolution changes for both partial and complete point cloud scans, 3) a planned grasp pose is executed with a mechanical gripper, and 4) grasp overlap is presented as a feature to identify regions on a partial object model ideal for object transfer or securing an object.

Keywords: Grasp Planning, Object Modelling, Mobile Manipulators, Next Best View, Point Cloud Processing, Unknown Environment, Grasping, Mechanical Grippers, Autonomous Grasping, Robot-to-Human Transfer

Dedication

To my parents (Lois and Darrol) and wife (Farnoush).

Acknowledgements

I extend my deepest appreciation and thanks to my senior supervisors, Dr. Kamal Gupta and Dr. Mehran Mehrandezh. They continuously offered sincere guidance and insightful advice throughout my PhD studies. Most importantly, they were always available if I had questions or needed help.

I would like to thank my family and friends for their constant support. I am grateful to my parents, Lois and Darrol, and wife, Farnoush, for supporting my educational journey and being beacons of positivity in my life.

To my fellow peers and students in the lab, thank you. I appreciate all advice and experiences shared about base navigation and tactile sensing throughout the years.

Table of Contents

List of Tables

List of Figures

List of Acronyms

CGAL	Computational Geometry Algorithms Library
CoM	Center of Mass
CR	Composite Rank (for NBV)
DOF	Degrees of Freedom
ICP	Iterative Closest Point Algorithm
IK	Inverse Kinematics
FC	Force Closure
FFT	Fast Fourier Transform
FFTW	Fastest Fourier Transform in the West
GICP	Generalized Iterative Closest Point Algorithm
GWS	Grasp Wrench Space
ML	Machine Learning
MSG	Messge
NBV	Next Best View
PCA	Principle Component Analysis
PC	Point Cloud
PCL	Point Cloud LIbrary
OI	Object of Interest
ROS	Robot Operating System
RRT	Rapidly-Exploring Random Trees
SDH	Schunk Dextrous Hand
SLAM	Simultaneous Localization and Mapping
TWS	Task Wrench Space

Nomeclature

{**A**}	Manipulator's Base Frame (or robot arm)
{**B**}	Mobile Base Frame
{$\mathbf{B}_m$}	Mobile Base Frame for object modelling
{$\mathbf{B}_g$}	Mobile Base Frame for object grasping
{**E**}	Eye-in-Hand Sensor Frame (i.e. Hokuyo laser scanner)
{**G**}	Grasp / Gripper Frame
{$\mathbf{G}_f$}	Final Grasp Frame
{$\mathbf{G}_p$}	Pre-Grasp Frame
{**O**}	Bounding Box Frame (surrounding objects)
{**W**}	World Frame
n	Normal Vector
p	Point Vector (or position)
T	Transform (or pose)
R	Rotation Matrix (or orientation)
M	Point Cloud Model (or image)
S	Point Cloud Scan
$\mathbf{H}_g$	Grasp Type / Gripper Histogram
$\mathbf{H}_o$	Object Histogram
$\mathbf{V}_g$	Grasp Type / Gripper Voxel Grid
$\mathbf{V}_o$	Object Voxel Grid
$_i$	i^{th} value or iteration
$_{org}$	Origin (of a frame, transform, or point)
$_{'}$	Prime (symbol) indicates a correction (of a frame, transform, or point)
$_{\_min/\_max}$	Minimum / Maximum value

Chapter 1.

Introduction

Autonomous mobile manipulation will be a central theme in the next generation of robotics applications. Mobile robots are anticipated to serve critical functions as helpers in homes, care centers, and varied circumstances where robots interact with humans. For instance, a survey in the health care industry ranked picking objects off the floor or shelf as the highest priority for people with disabilities[1]. Object retrieval, holding a tool, and opening a door are essential tasks examples for self-care and completing activities of daily living. A mobile robot retrieving medicine (or objects) could reduce the need for caregivers' assistance. For this to occur, autonomous systems need to be developed that combine perception and mapping, navigation, motion planning, and grasp planning. Our aim is to develop an autonomous system for a mobile robot to safely navigate an unknown environment, model unknown objects, discover available grasps on these objects, and execute different grasps depending on the task (i.e. fetch, carry, holding a tool, etc.) requested.

1.1. Related Works

1.1.1. Autonomous Mobile Manipulators

This research builds from and adds to previous work from the Robotics, Algorithms, an Motion Planning (RAMP) Lab to autonomously model an unknown object [2] and navigate to it in an unknown environment [3]. The key difference from [2] is that our modelling next best view (NBV) algorithm uses scan overlap as a key feature to improve model reconstruction. Although model reconstruction is performed by [2], merging and precisely registering scans is not explicitly evaluated; grasping is not involved. In [3], the object geometry, grasp location, and a grasp plan is known a priori. This presented work adds a significant capability by autonomously determining grasp poses for unknown objects by scanning, building a point cloud model, and performing grasp execution.

Industry is investing tremendously to automate an object picking task. For example, the Amazon Picking Challenge, formerly known as the Amazon Robotics Challenge, was sponsored yearly until 2018 to pick objects from a shelf and place them into a bin. These

challenges promoted suction-based or clasp-base grippers to pick known objects in a free environment [4]. However, an object dataset and layout of the environment is given to contestants a priori [4, 5]. A key difference to our work is grasping is completed with no object or environmental information known a priori. Grasping is accomplished without a database for multiple grasp types.

Two works present a grasping system for autonomous mobile manipulators, but their research solely focuses on grasping; issues associated to mobile manipulators, like base pose uncertainty, collision avoidance, and trajectory planning for the base and manipulator are not discussed [6, 7]. The grasping system presented in [6] uses stereo cameras to model an object and created a cost function to discover object surfaces that specifically fit their parallel jaw gripper. Modelling error was reduced by averaging surface normal within a voxel [6]. Work from [7] does not model objects; instead, their grasping system matches shape primitives to observed objects and executes known a prior grasps defined for each shape primitive. For mobile manipulators, no level of competence in research integrates autonomous object modelling with object grasping as a whole system. Work we present autonomously models an object until suitable grasp poses for multiple grasp types are discovered. Common issues that affect a mobile manipulators, like safe planning and experiencing base pose uncertainty, are addressed. Safe planning strategies for the base and manipulator are provided to navigate to scan viewpoints and final grasp poses outside the system's dextrous workspace.

1.1.2. Object Modelling

Within the field of data-driven grasping applied to unknown objects, two surveys published by Bohg et al[8] and Lei et al[9] review grasp approaches in great detail. In general, grasp strategies are executed in three sequential phases: i) Building an Object Model (i.e. Object Modelling), ii) Grasp Planning/Analysis, and iii) Grasp Execution[8, 9]. *Typically, the object modelling phase does not receive feedback from grasp planning to continue modelling.* Object modelling often assumes enough information is collected for grasp planning. Grasp strategies are usually demonstrated with fixed base manipulators, and as a result, key issues are avoided: i) assuming an eye-in-hand sensor, virtually no base pose uncertainties are associated with the fixed base case; hence, integrating scans from multiple viewpoints does not encounter problems associated with relatively large base pose uncertainty in our case, ii) grasping outside the fixed base manipulator's

workspace is not possible, and iii) safe planning (i.e. avoiding obstacles and unexplored space) is simplified by assuming a region in the manipulator's workspace is free; planning is limited to a manipulator's workspace instead of the much larger environment of the mobile manipulator.

From a systems point of view, most works within data driven grasping decouples object modelling from grasp planning. This implicitly treats a grasping task as an open-loop system that assumes enough features exist in an object model to successfully plan a grasp. Early examples of grasping an unknown object are demonstrated by Wang et al[10] and Bone et al[11]. In these works, the modelling phase observes an object at pre-determined viewing locations to capture a nearly complete model of an object. Object modelling is not autonomous and scan registration does not consider uncertainty. Spatial features have been extracted from household objects from a single viewpoint to be classified into several different object primitives for grasping [7]. Uncertainty is not addressed since it deals with a single scan, and hence, no scans are merged into a single model; instead, a complete primitive shape is assumed from a partial model. A gripper's pre-grasp shape (or end effector configuration) is used to fit a gripper along the surface of a partially complete point cloud[12, 13]. Works from [14-16] search for cylindrical or c-shape grasps that fit the point cloud model. Successful grasps from a single viewpoint have been demonstrated by extracting an object's centroid and orientation using principal component analysis (PCA)[17]. For these data driven methods, object modelling leads to successful grasps only if the viewpoint contains enough information to extract a grasp feature; further reconstructing the object until enough grasp features exist (and issues with this approach) are not directly addressed. A system should continue object modelling until a grasp is achievable.

Our proposed NBV modelling algorithm guides an eye-in-hand laser, mounted on a mobile base, to take scans of objects from multiple view poses with uncertainty and precisely register point cloud scans to build an object model. Most NBV algorithms, whether applying volumetric or surface variations, strive to guarantee consecutive scan overlap with previous ones, reveal new scan information, register/integrate scans with each other, and self-terminate when modelling is complete[2, 18-22]. Few algorithms are demonstrated with large uncertainty; a manipulator is generally fixed on a base in the world frame, and as a result, scans from the manipulator's sensor frame can be resolved with precision[23, 24]. Vasquez-Gomez et al introduced an NBV algorithm to address

position uncertainty somewhat indirectly by incorporating a distance metric that penalizes travel distance, and thus, reduces joint position uncertainty [25-27]. However, their algorithm is primarily demonstrated through simulation. Vasquez-Gomez et al did demonstrate their entire system using a real experiment and corrected odometry between scans by assuming the initial object location was known[26]. Although work from [22] compares [25] to others NBV algorithms with an experiment, position uncertainty is not directly assessed. In [24, 28], NBV algorithms generate viewpoints from an octree. Work from [24] adjusts a camera pose to avoid dynamics occlusions, while [28] models without a bounding box around objects and avoids obstacles along a ground plane. However, to aid simultaneous localization and mapping (SLAM) and reduce uncertainty, [28] used coloured tape as landmarks in their scene.

1.1.3. Analytical Grasping Methods

Analytical grasping approaches tend to apply specific grasp types that use two to four contact points. Some earlier works that present data-driven grasping from a point cloud applied force closure to determine grasp quality [10, 11, 29]. These works improved the computation time for force closure analysis by assuming a gripper pre-shape or introduced a heuristic stage that avoids searching for all contact combinations. For example, [29] showed three contact points form a triangular plane that can be decomposed to a series of IF-THEN statements, using contact normals, to determine a force closure grasp. The grasp planner presented in [10] reduced computation time for force closure by adding a pose constraint to align the gripper's palm with the object so all fingers will touch the object simultaneously when closed. The planner presented in [11] reduces computational time by searching for a parallel grasp; the grasp wrench would only be estimated for contacts that geometrically matched a parallel jaw. All of these works demonstrated grasping but these grasps are specialized and represent only one grasp type (for one specific purpose).

Later works generalize force closure to n-contacts, but experiments are demonstrated with randomly generated contacts or heuristics to consider a gripper's physical shape and constraints to match contact locations [30-32]. In contrast, our proposed grasping algorithm discovers potential grasp locations for n-contacts, multiple grasp types, and prioritized grasp poses for different tasks while satisfying a gripper's physical constraints.

1.1.4. Geometric Shape based Grasping Methods

Spatial features have been extracted from household objects from a single viewpoint to be classified into several different object primitives for grasping [7]. An object primitive heuristically simplifies an object's shape and represents it with a priori known geometric shapes (e.g. cube, cylinder, or sphere) [33, 34]. An object is either represented with a single primitive or can be decomposed into a group of sub-primitives [35-37]. Some learning-based methods also explore primitive shapes and discover grasps using simplified shapes [38-40] or similar objects [41, 42]. From these representations, either an analytical or data-driven database method can identify grasp locations. Comparably, object primitives can be viewed as a low resolution, quantized voxel grid setting applied by our algorithm that form similar cubic shapes. A key difference is our algorithm does not have prior geometric assumptions for an object model; their shape primitive can be thought of as a special case of our algorithm that occurs when using low voxel resolution to represent an object. An object's shape is not the key methodology we apply for grasping—representing the gripper shape and all corresponding grasp types is key.

1.1.5. Machine Learning and Heuristic Grasping Methods

More recently, machine-learning (ML) based approaches to grasp an object are presented in research. Many ML based grasp planning systems plan one grasp type per object (or object shape) [8, 9, 42-48]. Recently, [43] demonstrated a learning-based approach able to perform grasps using two grasp types. This algorithm did not select which type is most appropriate but demonstrated their framework can flexibly learn different grasp types. Another method selects between suction and pinch grasp modalities to retrieve objects from a container [49]. An approach, generalized to one grasp type, successfully grasps partial point cloud represented objects utilizing a convolution neural network trained with box-like and cylinder-like objects[46]. Force closure analysis measured grasp quality, and although not necessary, force closure is used to prune grasp pose results[46]. A key challenge for this approach, and similar grasp template approaches, is generating synthetic data to further classify more objects (i.e. sphere, toroid, cone, etc.).

Through human demonstration, earlier works demonstrated different grasp types to grasp a single object [50]. Even earlier work demonstrated database driven grasping for

approximately 7256 objects associated with 238,737 grasps for several different grippers [51]. These methods require large databases or significant training, which takes time to develop, and similar data-driven approaches do not scale well with respect to changes. For example, adding a new gripper or grasp type would significantly increase a database's size, and the grasp type added may become more difficult to discern from others already embedded.

In contrast, our proposed grasping algorithm demonstrates that different grasp types can be discovered in near real-time using conventional signal processing techniques, with no training or large databases.

1.2. Contributions

Key contributions to this research are:

1. Object Modelling and Grasp Planning phases are integrated via feedback to fully automate grasping from a potentially incomplete point cloud model. Furthermore, during an OI's point cloud model construction
    a. scan overlap is used as a key feature to mitigate registration error and guaranteeing correspondence between scans,
    b. registration correspondence for any iterative closest point (ICP) algorithm is improved by concatenating a 3D point with its respective 3D normal
2. Grasp planning is completely data-driven and can discover different grasp types from a partial scanned object with no offline-training.
    a. Added advantages are collision checks and planning a linear Cartesian trajectory for a grasp pose are easily integrated within the correlation step
3. A new grasping approach is introduced by representing grasp types and partially scanned objects using surface normal histograms and voxel grids
    a. Grasping is scalable for n-contact points, various grasp types, and is invariant to point cloud size
    b. Grasp pose solutions are found for several unique grasp types
4. Grasp selection changes based on the task. Grasp selection can change whether the task is defined as robot-to-human transfer or securing an object.

The material presented in this chapter is excerpted, reproduced, and modified with permission from the following papers: [52, 53].

M. J. Hegedus, K. Gupta, and M. Mehrandezh, "Towards an Integrated Autonomous Data-Driven Grasping System with a Mobile Manipulator," *IEEE Int. Conf. on Robotics and Automation (ICRA),* pp. 1596-1600, May 20-24, 2019[52]

M. J. Davari, M. J. Hegedus, K. Gupta, and M. Mehrandezh, "Identifying multiple interaction events from tactile data during robot-human object transfer," *28th IEEE Int. Conf. on Robot and Human Interactive Communication (RO-MAN)*, pp. 1-6, Oct. 14-18, 2019.[53]

1.3. Thesis Overview

This thesis is organized as follows:

- **Chapter 1** introduces the motivation and proposed work. Related works and expected research contributions are presented.

- **Chapter 2** presents the formal problem statement, mobile manipulator platform, and system-level solution for the problem statement. All discussion is high-level and focuses on the “world view” for planning, notation, and perception. Planning focuses to generate collision-free plans for the manipulator and base to either reach an NBV or final grasp pose. Notation introduces the mobile robot coordinate system and definitions used in the following chapters. Perception explains how all sensors scan the environment and update information planning.

- **Chapter 3** presents a detailed approach to generate a point cloud model for unknown object(s) using a next best view (NBV) algorithm.

- **Chapter 4** presents a detailed approach to generate grasp poses for multiple grasp types from a partial point cloud model. This section presents a method that generalizes grasp pose generation for mechanical grippers using partial point clouds.

- **Chapter 5** discusses conclusions and future work.

Chapter 2.

Autonomous Motion Planning System

2.1. Problem Statement

This work's purpose is to automate a mobile manipulator to autonomously model and grasp an unknown object within an unknown environment for an upcoming fetch and robot-to-human transfer task. From either a partial or complete object model, grasp poses representing various grasp types (i.e. pinch/lateral, tripodal, or power) are appropriately determined to grasp and lift an object of interest (OI). This system avoids collisions with obstacles and unobserved regions in the environment and assumes unknown object(s) to be grasped are at rest, can move freely, and located on a table within a known bounded region. Initially, the mobile robot faces towards the OI; a free path to the OI must exist to reach a proposed grasp pose. Simultaneous Localization and Mapping (SLAM) builds a map the mobile system navigates within[54], and assumes the environment is generally static (i.e. the system restarts if the OI location changes). Objects modelled and grasped are rigid and observable. Grasping actions assume the object's context (or purpose) is unknown. For example, this system cannot determine if an object is too heavy, fused to a table, or slippery; the system will naively grasp any location it determines is graspable using pre-defined grasp types.

2.2. Mobile Manipulator System

Our system hardware (i.e.Figure 2.1) comprises of a 3-DOF base (Powerbot), 6-DOF manipulator (Schunk PowerCube arm), and 7-DOF 3-fingered Schunk Dextrous Hand (SDH). One 3D and two 2D laser scanners are used for sensing. A Velodyne HDL-32E performs 3D environmental scans to detect free space for arm navigation; it is mounted 1.80m high above the base footprint frame (i.e. robot base's center). A SICK LMS100 mounted above the Powerbot's front bumper is used for base localization and mapping (i.e. 2D SLAM), and a Hokuyo URG-04LX is mounted on the manipulator's wrist as an eye-in-hand sensor. This sensor models objects and detects free space in the environment. Optical encoders estimate odometry from the drivetrain. Gmapping[54] implements SLAM because no inertial measuring unit (IMU) is installed in the Powerbot.

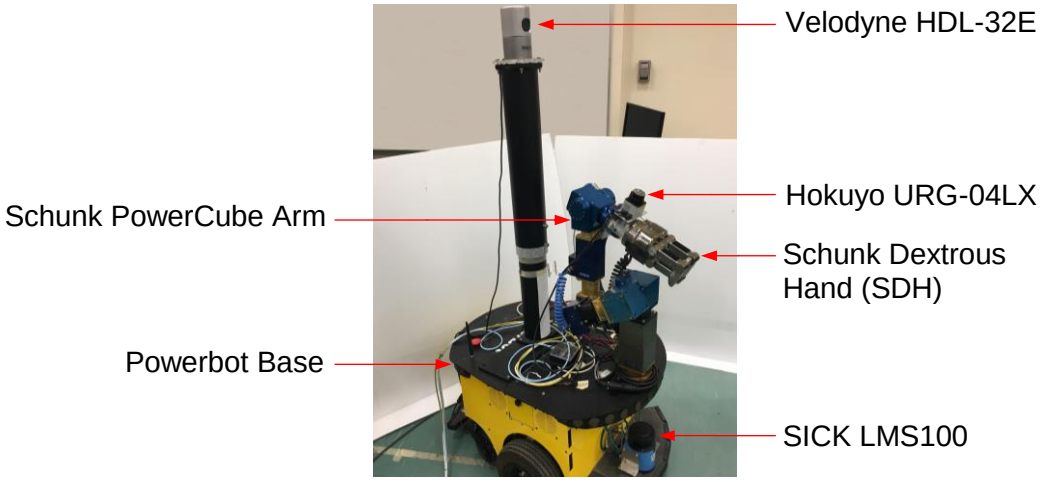


Figure 2.1: Powerbot Sensors & Gripper in a Folded Configuration.

2.3. Base Pose Uncertainty Problem

Mobile manipulator systems are susceptible to base pose uncertainty (i.e. the robot is not located at its assumed base pose). In a real-life system, estimating and reaching a base pose is affected by three factors: 1) 2D map quality generated from SLAM, 2) base pose localization estimated by SLAM, and 3) navigation minimizing error to reach its desired base pose goal (controlled using pose tolerances).

2.4. System Overview

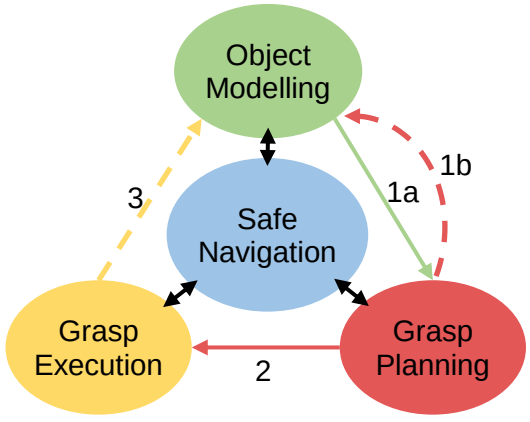


Figure 2.2: Three Phase Methodology to Grasp Planning

A high-level overview for the proposed grasping system is shown in Figure 2.2. In this figure, nodes represent stages while numbered branches indicate the system progression to grasp an unknown object; solid branches represent success while dashed branches

represent failure from the previous stage. Grasping an object is accomplished in three major stages: 1) **Object Modelling** creates a point cloud representation for all objects being scanned within a bounded region, 2) **Grasp Planning** determines if the object can be grasped, and 3) **Grasp Execution** discovers a base pose that allows the manipulator to reach and execute a collision-free grasp; this phase corrects base pose error that affects guiding the gripper to its final goal. All stages utilize Safe Navigation. **Safe Navigation** is unique because all perception focuses to analyze the global environment to determine base poses that are collision-free for both the mobile base and manipulator. In contrast, perception from the other stages focus to analyze object(s) within a local bounding box.

A key feature for this grasping system is Grasp Planning is integrated with Object Modelling. Grasp Planning provides feedback to Object Modelling to temporarily pause modelling if a grasp exists. Grasp Execution is needed to correct base pose uncertainty; as the robot travels, base pose error propagates to the desired grasp pose in the world frame. Prior to executing a grasp, base pose uncertainty is estimated to correct proposed grasp poses to allow the gripper to reach its desired grasp pose.

2.5. System Detailed Approach

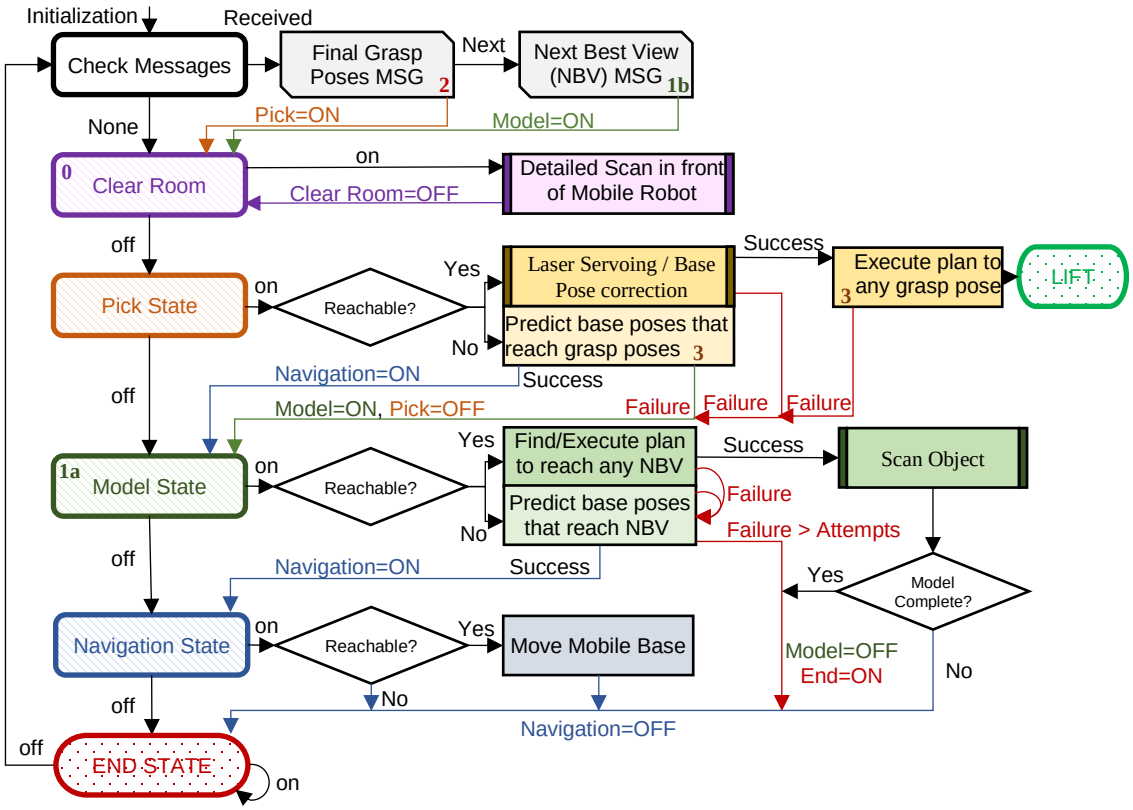


Figure 2.3: System State Diagram

The methodology shown in Figure 2.2 is implemented using a state machine presented in Figure 2.3. Numbered branches in each figure correspond to each other and generally indicate the system's progression in chronological order. States presented on the left are ordered in terms of decreasing priority (i.e. END State is lowest priority). Not shown, are two sub-systems, detailed in later chapters, which determine the next best views for Object Modelling and final grasp poses for Grasp Execution; the output from these sub-systems are represented as messages (MSGs) in Figure 2.3 to trigger Model State and Pick State within the state machine. From a system perspective, these messages represent the NBV to scan the object(s) of interest (or OI), and final grasp pose to grasp any object. The overall system (or state machine) is responsible for collision-free planning to reach these positions if they are reachable.

2.5.1. Clear Room State Overview

The system begins with necessary initial conditions (i.e. Clear Room and Model State active). Clear Room state actively scans the front of the mobile robot in detail using all available laser scanners to observe free space within the environment for safe navigation; any observed obstacle or unobserved space is treated as an obstacle to avoid. This step is necessary to create and initialize all maps used by navigation.

2.5.2. Model State Overview

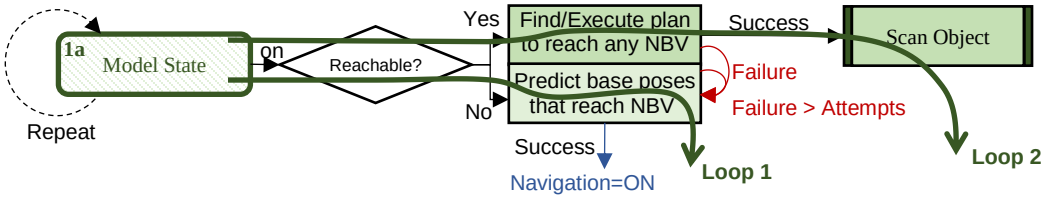


Figure 2.4: Model State Diagram

After Clear Room state completes, Model State checks if a list of NBVs are inside the robot's dexterous workspace. If any NBVs are potentially reachable without moving the base, a collision-free plan for the manipulator is calculated; otherwise, the system randomly samples base poses underneath NBV locations. At proposed base pose locations, collision-free plans for the manipulator are estimated for the list of NBVs within the robot's workspace, and if a collision-free plan exists, the mobile base navigates to that proposed base pose. If no plan is found, the next proposed base pose is sampled, and this process repeats until a successful plan is predicted for a predicted base pose.

Two loops, shown in Figure 2.4, are needed to pass through Model State to execute an object scan. The first loop triggers a base pose prediction that permits the manipulator to reach an NBV. Navigation State is activated, the mobile base moves to the base pose prediction, Navigation State is de-activated, and the second Model State loop begins. Since NBVs are now within the manipulators dexterous workspace, a collision-free plan for the manipulator is estimated for all reachable NBVs. If no valid plans are found (e.g. the manipulator is blocked by an obstacle), base pose prediction is re-triggered, and another Model State loop is repeated.

2.5.3. Pick State Overview

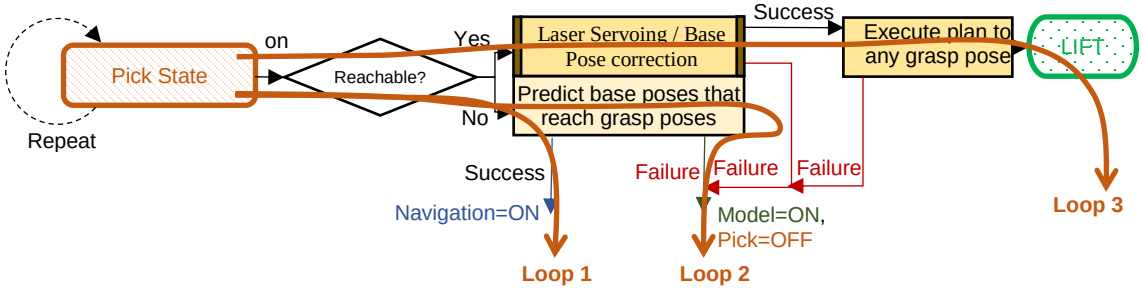


Figure 2.5: Pick State Diagram

Pick State activates when grasp poses to secure an object are discovered. A Final Grasp Pose MSG is received, and this message contains two candidate lists: 1) a grasp type (i.e. lateral, tripodal, or power) and 2) corresponding grasp poses to be attempted on a partial object model. As shown in Figure 2.3, Pick State is prioritized over Model State when active. Similar to Model State, grasp poses are executed if they are reachable and exist within the manipulators dextrous workspace; if not, base poses are randomly sampled around the modelled object(s), and collision-free plans for the manipulator are esimated to reach any grasp pose from a sampled base pose. If a collision-free plan exists, the mobile base navigates to the sampled base pose (i.e. Figure 2.5 Loop 1). If no plan is found, Pick State is disabled, and the system continues object modelling (i.e. Figure 2.5 Loop 2).

After the mobile base navigates to a new base pose, candidate grasp poses are within its dexterous workspace. Pick State is repeated a second time to grasp and lift an object. Lifting an object is executed in three consecutive steps. First, the eye-in-hand laser scans the object(s) again (i.e. laser servoing) and corrects the current base pose relative to the observed object(s). Second, a collision-free plan for the manipulator to reach any grasp pose is estimated again because the current base pose changes after correction; grasp poses may no longer be reachable due to how much base pose uncertainty affects the system. If a collision-free plan exists, it is executed, and the gripper uniformly closes all fingers at the final grasp pose to securely grasp an object. Thirdly, the manipulator lifts the object vertically to show a secure grasp (i.e. Figure 2.5 Loop 3). If any failure (i.e. laser servoing, trajectory planning, or trajectory execution) occurs during these three steps, the system aborts, disables Pick State, and continues object modelling (i.e. Figure 2.5 Loop 2).

Failures are typically caused after base pose correction; the mobile base may no longer be at its ideal predicted position and is too close (or far) to discover a new collision-free plan to reach the final grasp pose. Theoretically, the mobile base could move to the corrected base pose, but this is impractical because base pose uncertainty is reintroduced while the mobile base navigates. Less commonly, the manipulator's trajectory may abort because joints do not precisely reach a waypoint within the planned trajectory, or an obstacle blocking the trajectory may be observed erroneously. For these cases, Pick State is aborted to not disturb any object.

2.5.4. Navigation State Overview

Navigation State receives a base pose list from Model or Pick State and navigates the mobile base to a collision-free base pose. Prior to navigation, the robot's manipulator is folded within the robot's base footprint to prevent it from colliding with obstacles. A base pose is accepted for navigation if both the base footprint and volume above are free (i.e. region is observed and no obstacles are observed). If all base poses are rejected, Navigation State is disabled and Model State is repeated to sample new base pose locations. A decoupled approach to navigation (i.e. manipulator and base move separately) is chosen to ease implementation and reduce computational cost.

2.6. Definitions and Notations

Coordinate frames are represented using Denavit-Hartenberg notation[55]. Initially, the world frame (**W**) and base frame (**B**) poses coalign. The base frame (**B**) is attached to the base footprint and is primarily used to navigate to either a base frame pose for object modelling ($\mathbf{B}_m$) or base frame pose for grasping ($\mathbf{B}_g$); subscripts 'm' and 'g' denote modelling and grasping respectively. A manipulator (arm) is located on top of the mobile base, and the manipulator's base frame (**A**) attaches to the manipulator's base, on top of the mobile base. Trajectory planning, prediction, and the manipulator's workspace is relative to the manipulator's base frame (**A**). Visually, this frame is seldom shown because it is an intermediate transform to reach the world frame where most information is shown and represented. An eye-in-hand Hokuyo sensor frame (**E**) is attached at the end of the manipulator, above the wrist, and is orientated coplanar to the laser scanner's field of view. The SDH gripper also connects to the end of the manipulator. A general

grasp/gripper frame (**G**) is attached between all SDH fingers, near between distal joint motors. The gripper frame (**G**) is meant to reach a pre-grasp frame ($\mathbf{G}_p$) and move towards the final grasp frame ($\mathbf{G}_f$) to grasp an object; subscripts 'p' an 'f' denote a pre-grasp and final grasp poses respectively.

Position ($\mathbf{p}, \mathbf{p} \in \mathbb{R}^{4\times1}$), rotation ($\mathbf{R} \in \mathbb{R}^{4\times4}$), translation ($\mathbf{t}, \mathbf{t} \in \mathbb{R}^{4\times1}$) and pose/transformation ($\mathbf{T} \in \mathbb{R}^{4\times4}$) use superscript and subscript annotation to designate the reference/designation and source coordinate frames respectively. Pose (**T**) is a homogenous transform, i.e. ${}^{A}_{B}\mathbf{T} = ({}^{A}_{B}\mathbf{R}, {}^{A}_{B}\mathbf{t})$. All goal poses are visually shown in the world frame (**W**).

2.7. World Octree ($\mathcal{W}$) for Planning and Collision Avoidance

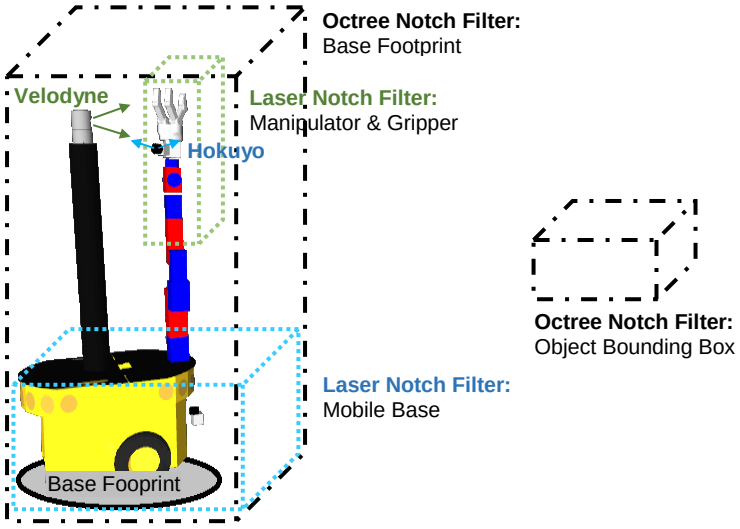


Figure 2.6: World Octree "Spatial" Notch Filters

For mapping and perception, an octomap octree encompasses the environment and mobile manipulator[56, 57]. An octree is chosen because its 8-array tree structure can efficiently represent an environment without consuming a large memory footprint[56]. An octomap implements an octree structure as a volumetric, probabilistic 3D occupancy map; tree leafs (or voxels) are updated by current laser range sensors combined with prior observations to assign a low or high probabilistic values to indicate a leaf is empty or occupied respectively. Probabilities are discussed using log-odds notation. Obstacles (or occupied leafs) are coarsely represented (i.e. low voxel resolution) for collision detection and avoidance. A key feature of this work is unobserved space is assumed to be an obstacle to either be scanned or avoided.

Initially, every leaf within the octree $\mathcal{W}$ is initialized as an obstacle. Shown in Figure 2.6, two filters, represented as dashdotted lines, free (i.e. clear) space within the octree $\mathcal{W}$. Each filter is kin to a “spatial” notch filter removes initialized obstacles within a bounded region, and as such, we refer to them as a notch filter. One notch filter frees all volumetric space above the mobile manipulator’s footprint to allow the manipulator limited, collision-free, movement. A second notch filter frees space within the bounding box surrounding the object of interest to be scanned; this second filter prevents scanned objects to be considered as obstacles and is necessary to position a gripper for grasping. After initialization, the octree $\mathcal{W}$ is updated by laser readings from the Velodyne sensor and Hokuyo eye-in-hand sensors. False-positive obstacles detection due to the lasers self-scanning the mobile manipulator is avoided by a third laser notch filter, represented as dotted lines; the Velodyne, shown as green, ignores any obstacles within a bounding box attached at manipulator’s wrist frame, and the Hokuyo eye-in-hand sensor, shown as blue, ignores any obstacles within a bounding box attached to the robot’s base frame.

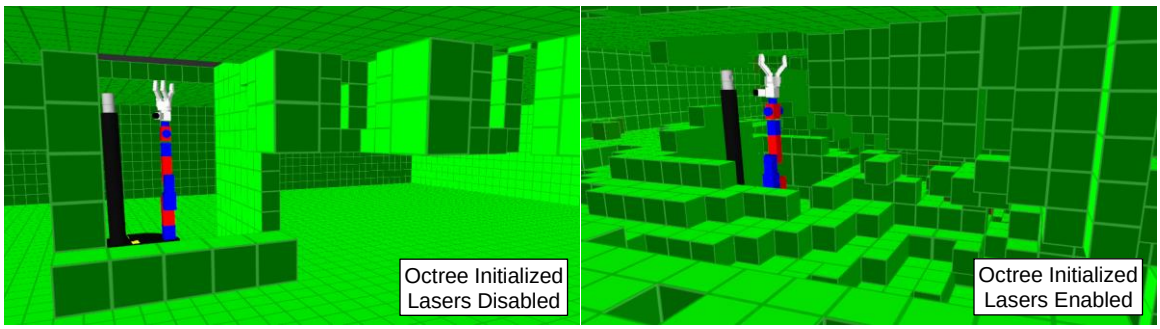


Figure 2.7: World Octree Obstacle Perception (Before & After Sensors are Enabled)

Figure 2.7 shows the octree $\mathcal{W}$ initialized by an octomap server[57] prior to enabling lidar sensors. Occupied octree leafs are represented as solid green voxels, and for visualization, only surfaces of solid occupied leafs are displayed. When the octree $\mathcal{W}$ is initialized, the mobile manipulator sits within a free region surrounded by occupied leafs. When lasers are enabled to update the octree $\mathcal{W}$, free regions are updated, but a ‘cone’ of obstacles surround the robot because this region is not observed by lidar sensors. Using pre-defined movements, the eye-in-hand sensor observes regions outside the beamwidth of the Velodyne sensor. While controlling the eye-in-hand sensor pose, the robot’s manipulator avoids collision with any occupied leaf in the octree $\mathcal{W}$.

Error! Reference source not found. summarizes how self-scanning is additionally mitigated by controlling when readings are accepted to update the world octree. In general, if the mobile base is stationary and the manipulator is homed, the Velodyne lidar is always on. The Velodyne is disabled when the manipulator moves, unless this movement is pre-defined (i.e. panning the wrist to scan an object or rotating the arm to scan the environment). If the mobile base is moving and manipulator is homed, both sensors are on; the Velodyne scans the entire room while the eye-in-hand sensor scans the floor in-front of the mobile robot. Due to the states defined in Figure 2.3, the mobile manipulator cannot move its base and arm simultaneously.

Table 2.1: Lidar Sensor Truth Table for World Octree

Base Stationary?	Manipulator Moving?	Movement Pre-Defined?	Velodyne Status	Eye-in-Hand Status	Comment
F	F	N/A	**ON**	**ON**	Navigation
F	T	X	**OFF**	**OFF**	Not Possible
T	F	N/A	**ON**	**OFF**	Navigation Ended
T	T	F	**OFF**	**OFF**	Moving Arm/Gripper
T	T	T	**ON**	**ON**	Scanning

2.8. Object Octree ($\mathcal{M}$) for Object Modelling

Modelling is guided using an additional higher resolution octomap[57] octree that fills object bounding box show in Figure 2.6. All object(s) surfaces are reconstructed within this bounded region, and octree probabilities will classify leafs as free, occupied, or unknown[18]. Initially, all object octree leafs are initialized as unknown. Afterwards, the object octree is updated using the eye-in-hand sensor by copying raw laser data from obstacle perception. An octomap is selected because probabilistic updates account for sensor noise and dynamic changes in the environment. Our environment is static, but odometry uncertainty propagates to a mobile manipulator's scanning frame; modelling while experiencing uncertainty can be viewed like changing the object pose between scans. Information captured by the last scan is important because it accurately localizes object features relative to the mobile manipulators base footprint frame. Information from those recent features can reconstruct an object model, even while a robot experiences large pose uncertainties.

2.8.1. Object Octomap-Specific Implementation

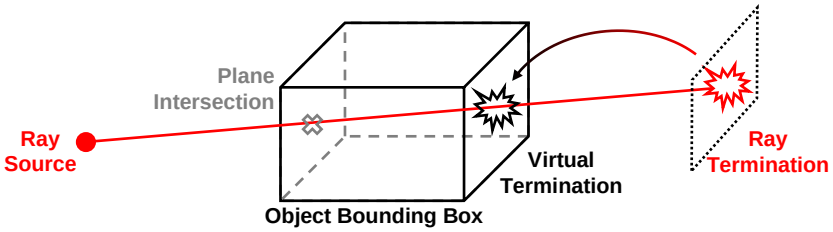


Figure 2.8: Ray-Cube Intersection Visualization

A ray-cube intersection algorithm detects when the eye-in-hand laser data passes through octree $\mathcal{M}$'s bounding box, and if this occurs, laser data is virtually terminated along the bounding box's border[58]. Shown in Figure 2.8, a ray intersects through two planes of a cube, and given the ray pose (known from odometry), a laser hit is projected along the plane furthest from the ray source. Practically, this step is necessary to update the octomap octree because its current implementation only updates if a laser 'hit' occurs within its defined bounds.

2.9. Base Pose Selection

2.9.1. Object Modelling Base Poses

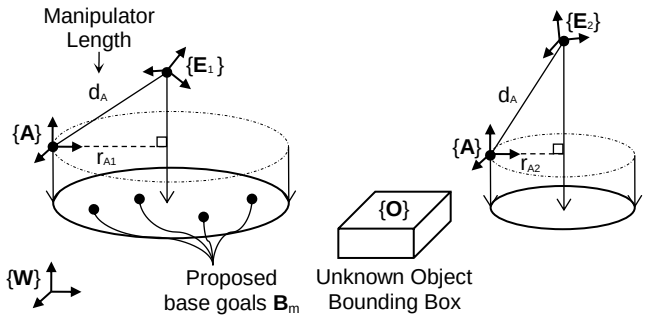


Figure 2.9: Object Modelling Base Pose Generation

Figure 2.9 visualizes object modelling base goal generation, denoted as $\mathbf{B}_m$, randomly sampled for any sensor scanning frame, denoted as {**E**}. Given the fully extended length of a manipulator's arm, denoted as d_A, a circle is projected onto the ground plane below {**E**}, shown as a solid circle. A parallel dashed circle is shown because the manipulator's base frame, denoted as {**A**}, is raised above the ground plane because it is attached onto a mobile base. Trigonometry determines radius r_A for every scanning pose **E** pointing towards the object bounding box, i.e. $\mathbf{E}_1$, $\mathbf{E}_2$, etc. Within each projected circle,

the manipulator can reach the goal pose if it is fully extended. A circle's radius changes size for each scanning pose; for example, the circle becomes smaller as the goal is elevated in the world frame because the manipulator needs to reach more vertically.

Proposed modelling base goal locations, $\mathbf{B}_m$, are randomly sampled within a projected circle's radius. The $\mathbf{B}_m$ frame orientation is orthogonal to the bounding box's during the modelling phase. To select a candidate $\mathbf{B}_m$, each scanning pose is projected (or transformed) into the current manipulator's frame using odometry, i.e. and inverse kinematics (IK) determines if the manipulator's sensor can reach frame $\mathbf{B}_m$. The following equation projects a scanning frame, i.e. {**E'**}, into the current manipulator frame {**A**} using the robot's current base frame {**B**} as an intermediate step:

$$ {}^{W}_{E_1'}\mathbf{T} = {}^{W}_{A}\mathbf{T}\,{}^{A}_{B}\mathbf{T}\,{}^{B}_{E_1'}\mathbf{T}\,, \qquad \text{where } {}^{B}_{E_1'}\mathbf{T} \equiv {}^{B_m}_{E_1}\mathbf{T} $$

Control for the base and manipulator are decoupled; typically, the base moves with the manipulator safely folded to a base pose goal for the manipulator to reach a scanning pose, and once the base reaches its goal, the arm moves the eye-in-hand sensor along a collision free trajectory from the folded configuration to scanning pose. To guarantee these actions are safe, the manipulator is constrained to execute trajectories within observed free regions within octree $\mathcal{W}$. Prior to moving to any candidate base goal, octree $\mathcal{W}$ is queried to determine if a bounded region around the manipulator is free at the candidate base goal within octree $\mathcal{W}$. If this condition is true, the base goal is selected and executed; while navigating, free space in the 2D costmap is assumed free for both the base and manipulator.

2.9.2. Grasping Base Poses

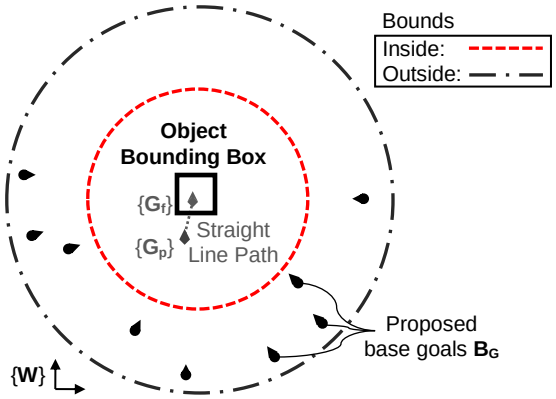


Figure 2.10: Grasping Base Pose Generation

Given a set of potential grasp poses, random base positions around the OI are sampled with the robot orientated facing the OI (i.e. the OI is 0° relative to the base frame). Samples are bounded between an inside and outside circle; the inside circle is greater than navigation's inflation radius[59, 60] (to prevent collision) and outside circle is less than the manipulator's dexterous workspace (to reach the final goal). For each candidate base pose, denoted as $\mathbf{B_G}$, the IK are numerically calculated to reach pre-grasp $\mathbf{G_p}$ and final grasp $\mathbf{G_f}$ poses (see Figure 2.10). If the IK to reach $\mathbf{G_p}$ and $\mathbf{G_f}$ are successful and a straight line path passes through each pose, the base pose's rank is incremented. Each base pose generated will check all final grasp poses, and a base pose that successfully reaches most final grasp poses will be ranked the highest; after ranking, the mobile robot moves to the base pose with the highest rank. If two base poses have the same rank, the first base pose evaluated is selected. The manipulator executes a trajectory to $\mathbf{G_p}$ with a closed gripper, and at $\mathbf{G}_p$, the gripper opens for a Cartesian planner to execute a forward trajectory to $\mathbf{G_f}$. If the manipulator cannot complete a trajectory to $\mathbf{G_p}$ or discover a Cartesian plan to $\mathbf{G_f}$, the system aborts, executes another model scan and attempts another grasp.

2.10. Evaluation from Implementation

During development, unrecorded system tests were performed to confirm navigation and perception; no explicit evaluations are recorded to report. ROS MoveIt!'s interface[61] is implemented for trajectory planning, and MoveIt!'s collision avoidance octomap[57], accessed by MoveIt!'s planning scene class, is replaced with octree $\mathcal{W}$. While integrating

perception, pre-defined trajectories swung the arm back and forth within octree $\mathcal{W}$'s observable free space. Afterwards, the trajectory is repeated with an obstacle inserted along the trajectory's planned path; motion execution failed. However, when the planner was allowed to replan, a new trajectory is discovered to move the manipulator around the obstacle to reach its final goal. In fact, Clear Room State describe in Section 2.5.1 is needed to initiate autonomous planning because the system is initially surrounded by obstacles and unobserved space.

Collision-models (i.e. defined to be slightly larger than physical dimensions) and threshold parameters (i.e. joint and goal tolerances) are selected to prevent the manipulator and base to collide into observable obstacles. These measures are confirmed through simulation and impolitely evaluated during experiments reported for object modelling and grasping objects using multiple grasp types, described in later chapters. Implicit knowledge confirms the mobile manipulator does not collide into obstacles, and the manipulator planner avoids collisions within octree $\mathcal{W}$[61, 62].

Chapter 3.

Integrating NBV Modelling with Grasping

3.1. Problem Statement

An autonomous, fully-integrated, data-driven planning system is proposed for a mobile-manipulator to grasp an unknown object in an unknown environment. Object modelling with a wrist mounted lidar sensor, using a next best view (NBV) algorithm, is integrated with grasp planning. The object of interest (OI) to be grasped is assumed rigid and inside a known bounded region in the world frame; no other a priori knowledge about the OI is assumed. Preliminary results is presented of a 9-DOF mobile manipulator (with a gripper) autonomously scanning and modelling an OI, and from the partially constructed model, a grasp is planned; if planning is successful, grasping the OI is executed, but otherwise, further scans are taken to build a more complete model until grasping the OI is achieved.

3.2. Motivation

A mobile manipulator provides more flexibility retrieving objects in a large workspace; however, errors associated with localizing the base can propagate to the sensor frame and adversely affect object modelling and grasp execution. Object modelling is affected because consecutive scans need to correctly register for integration. This action is typically performed using an Iterative Closest Point (ICP) algorithm[63]. ICP performs well when the initial guess is accurate but can fail in the presence of large uncertainties. A mobile manipulator's registration error (i.e. distance between two overlapping scan) can be larger than the width of the scanned object. Generally, NBV algorithms that consider registration are not demonstrated to correct this issue[20, 21, 25, 27]. An algorithm adaptation is necessary to treat scan overlap as a primary feature to guarantee registration correspondence. Another key aspect of this work is our NBV algorithm guides the eye-in-hand laser, mounted on a mobile base, to take scans of an OI from multiple view poses with uncertainty and precisely register point cloud scans to build an OI model.

3.3. NBV Modelling System Integration

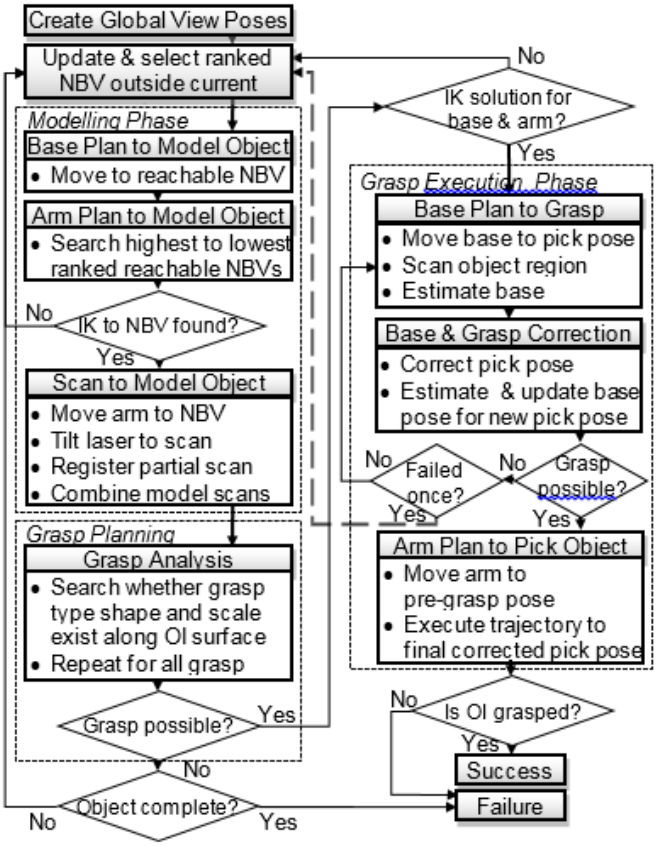


Figure 3.1: Flow Diagram to Integrate Modelling with Grasping.
This figure is reprinted with permission from [52]

As discussed in Section 2.4, Object Modelling is integrated with Grasp Planning. A high level flowchart of our integrated and autonomous system to grasp an OI is shown in Figure 3.1. Initially, the robot begins the *Modelling Phase*; in this phase, a volumetric NBV algorithm (see [22, 64]) evaluates and ranks potential view poses that surround the OI. Shown in Figure 3.2, a global list of uniform randomly sampled view poses, within an inner and outer ring around the bounding box, encompassing the OI is first created and then ranked. Heuristically, a cylindrical shape is chosen to approximate a toroid, representing the robot manipulator's workspace if the base moves circularly around the OI. If a portion of the ring is blocked by an obstacle, a mobile manipulator can travel to other free locations to continue OI modelling. View poses surrounding the OI ensure scans cover all sides of the object. The inside and outside ring radius provides depth for the mobile manipulator to select a range of NBVs in case several are unreachable due to obstacles. Ring height is similar to the manipulator's dextrous workspace height.

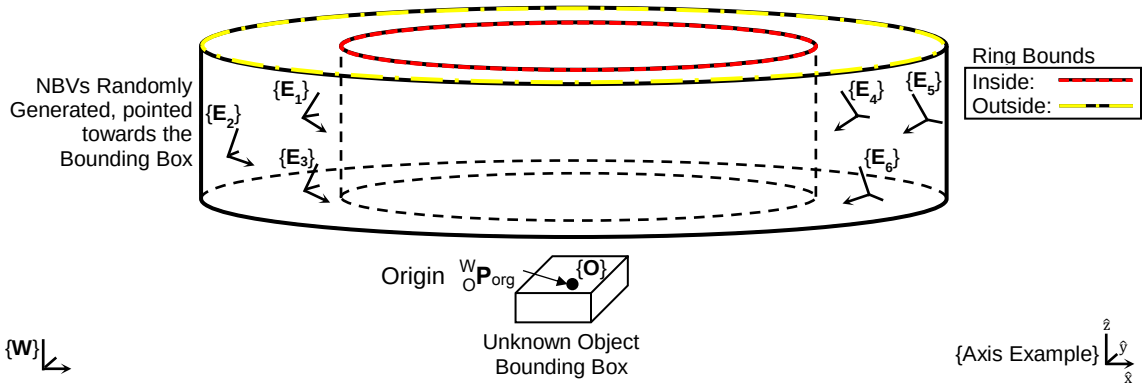


Figure 3.2: NBV Frame Generation

NBV orientation, shown as frames in Figure 3.2, is represented by three vectors. Frame vector axes are defined for world frame, **W**, as ${}^{W}\hat{\mathbf{x}}$, ${}^{W}\hat{\mathbf{y}}$, ${}^{W}\hat{\mathbf{z}} \in \mathbb{R}^{3\times1}$ and sensor (or NBV) frame, **E**, as ${}^{E}\hat{\mathbf{x}}$, ${}^{E}\hat{\mathbf{y}}$, ${}^{E}\hat{\mathbf{z}} \in \mathbb{R}^{3\times1}$. A frame is attached to the object's bounding box center, denoted as {**O**}. Initially, a NBV's x-axis is created by normalizing the difference between the bounding box's origin, i.e. ${}^{W}_{O}\mathbf{P}_{org}$, and NBV frame's origin, i.e. ${}^{W}_{E}\mathbf{P}_{org}$. This vector represents the sensor frame facing towards the OI. Next, the NBV's y-axis is created by taking the cross-product between the NBV's x-axis and world frame's z-axis, and the cross-product between the NBV's x-axis and y-axis. These steps are summarized below:

$$ {}^{E}\hat{\mathbf{x}} = \frac{{}^{W}_{O}\mathbf{P}_{org} - {}^{W}_{E}\mathbf{P}_{org}}{\left\| {}^{W}_{O}\mathbf{P}_{org} - {}^{W}_{E}\mathbf{P}_{org} \right\|} $$

$$ {}^{E}\hat{\mathbf{y}} = {}^{E}\hat{\mathbf{x}} \times {}^{W}\hat{\mathbf{z}} $$

$$ {}^{E}\hat{\mathbf{z}} = {}^{E}\hat{\mathbf{x}} \times {}^{E}\hat{\mathbf{y}} $$

The highest ranked view pose (or NBV) is selected to scan the OI (see 'NBV Ranking' for ranking criteria). For planning motions to reach the NBV, the Hierarchical and Adaptive Mobile-manipulator Planner (HAMP) [3] would be ideal; however, we chose a decoupled approach to ease implementation and reduce computational cost. Our approach works as follows. The base plans within a 2D costmap[59], and at the base's goal, the arm plans within a global 3D octree [57] using Rapidly-exploring random trees (RRT)[65].

Base poses, described in Section 2.9.1, are randomly sampled around the region of base poses from where the NBV pose is reachable. Each sampled base pose is tested to determine if a trajectory exists for the manipulator to reach the NBV. The robot moves

to the first available base pose if its location is collision free for both the base and arm. Once the robot arrives at its base pose, the eye-in-hand sensor moves to the highest ranked NBV; if this is unreachable (e.g. a new obstacle is discovered or the base stops offset from its desired pose), the next highest rank NBV is chosen. Once at the NBV, the eye-in-hand sensor initiates a scan. After a scan, three actions occur: i) the partial point cloud representation of the OI is updated using a generalized iterative closest point (GICP) algorithm[66], ii) all NBV rankings are updated, and iii) *Grasp Planning/Analysis* discovers if a gripper shape fits along the partial OI model for all grasp types. If grasp analysis discovers a match and generates grasp poses around the point cloud model, grasp pose(s) are sent to the robot, and the robot initiates the *Grasp Execution Phase*; otherwise, the robot moves to a new location for scanning, and the process repeats itself until the model is complete. The number of contact points associated to the group is determined by the grasp type attempted (e.g. two contact points for a pincer/parallel grasp, three points for a tripod grasp, etc.). During the *Grasp Execution Phase*, the manipulator moves into a pre-grasp position. From there, a Cartesian planner moves the end effector along a straight line trajectory from the pre-grasp to final grasp pose.

3.4. NBV Object Modelling with Uncertainty

The NBV modelling algorithm is frontier-based (i.e. a frontier is the boundary between free and unknown regions) and uses scan overlap as a key feature to reduce uncertainty. It is inspired by existing methods with adaptions to emphasize scan overlap from the previous scans[25, 64]. Scan overlap is used as a feature to mitigate registration error because it guarantees a specific ratio of correspondence for registration between consecutive scans. This overlap ratio is adjustable to improve registration accuracy (i.e. more scan overlap when more uncertainty exists). A GICP algorithm is used to merge current with previous scans, and in effect, GICP corrects position uncertainty. This algorithm is not optimized to fill details in the model or scan small occlusions. Our goal is to practically scan large, visible, and easily reachable surface features so the OI can be quickly and easily grasped. Constraints, i.e. field of view and viewing angle, are incorporated by ray casting from the sensor's pose towards octree $\mathcal{M}$'s bounding box. Indirectly, occlusions within the bounding box are revealed by selecting view poses that overlap information gained by the previous scan. View poses

that are obstructed by obstacles and do not have line-of-site to the OI within octree $\mathcal{W}$ are also ignored.

3.4.1. NBV Volumetric Representation

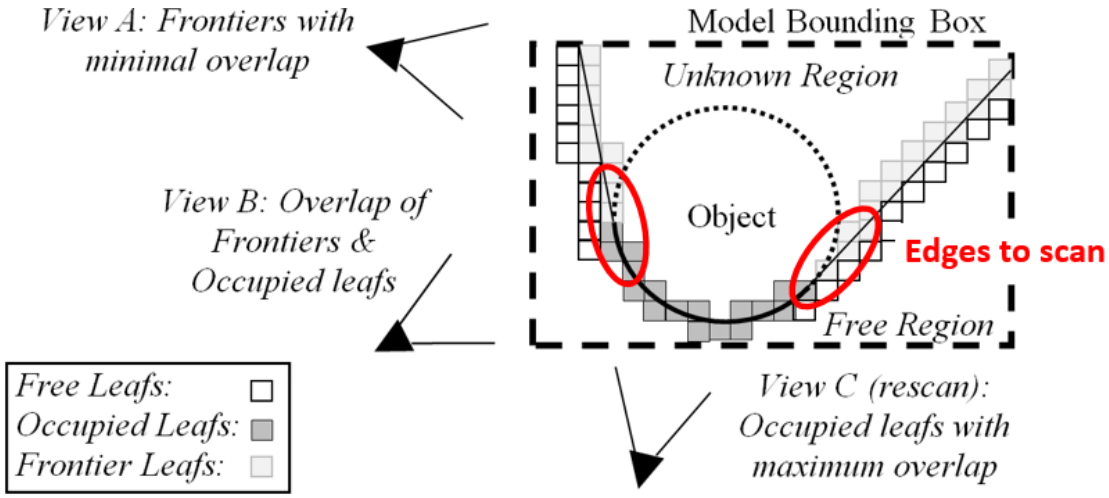


Figure 3.3: Next Best View (NBV) Octree Representation.
This figure is reprinted with permission from [52]

Selecting viewpoints is accomplished by defining a bounding box around the OI and representing it with an octree [57] to guide the NBV algorithm. Sensor data updates these leafs using a Bayes filter to build a probabilistic occupancy map[57]. We assign four states to represent leafs: i) free, ii) occupied, iii) unknown, and iv) frontier. They are defined below, and a 2D representation is shown in Figure 3.3.

- A leaf is ***free*** if its occupancy is below a threshold, p_{min}
- A leaf is ***occupied*** if its occupancy is above a threshold, p_{max}
- A leaf is ***unknown*** if it is either not scanned, or it is scanned but contains an occupancy probability value between p_{min} and p_{max}
- A ***frontier*** leaf is any *unknown* leaf adjacent to a *free* leaf's faces

3.4.2. NBV Ranking

NBVs are ranked by comparing a percentage of occupied and frontier leafs within octree $\mathcal{M}$. A positive weighting factor ω, ω ∈[0,1], creates a vector $\mathbf{w_d}$=(ω,1- ω) to represent the desired overlap percentage for each scan. Let η_{OCi} and η_{FRi} represent the occupied count and frontier count respectively for the i^{th} viewpoint. η_{OCi} and η_{FRi} are derived for each viewpoint by casting rays within the sensor's field of view within the octomap; if a ray

terminates at either an occupied or frontier leaf, η_{OCi} or η_{FRi} will be incremented respectively. Assuming a finite total of 'n' NBVs, where i={1, 2, …, n}, we define another vector $\mathbf{w_i}$, whose components are the fractions of frontiers and occupied leafs observed at the i^{th} viewpoint is:

$$\mathbf{w}_i = \left(\frac{\eta_{FRi}}{\max(\eta_{FR} \mid n)} \ , \ \frac{\eta_{OCi}}{\max(\eta_{OC} \mid n)} \right)$$

$$CR_i = \begin{cases} \dfrac{\mathbf{w_d}}{|\mathbf{w_d}|} \cdot \dfrac{\mathbf{w}_i}{|\mathbf{w}_i|}, & \eta_{FRi} \ , \ \eta_{OCi} > \text{thresh} \\ 0 \ , & \text{otherwise} \end{cases}$$

An NBV's composite rank (CR) is given by the dot product of normalized $\mathbf{w_d}$ and $\mathbf{w_i}$ vectors. CR scales between [0, 1] and is maximal when $\mathbf{w_i}$ equals the desired overlap $\mathbf{w_d}$. Each rank represents how similar a viewpoint $\mathbf{a}_i$ matches a desired overlapping ratio $\mathbf{w_d}$, and its value decreases as a view pose moves further away from a maximal CR. If either η_{OCi} or η_{FRi} are below a minimal threshold, the NBV rank is set to zero due to not enough information contributing to the CR. Ranking criterion purposely has an intuitive and practical meaning for the user. For example, if ω = 0.5 (i.e. View B in Figure 3.3), ranks favour an equal ratio of occupied and frontier leafs. As ω decreases, more overlap is preferred.

Heuristically, CR is automatically set to zero if a ray cast from a view pose does not terminate at an occupied or frontier leaf, leafs required to estimate CR. This heuristic ignores sampled view poses that are occluded by world obstacles or view poses that do not observe a leaf type needed to calculate CR. To ignore world obstacles, one ray cast from a sampled view pose to OI is performed within octree $\mathcal{W}$, and if the ray cast terminates at any obstacle leaf before reaching the OI, the view pose is occluded by an obstacle (and CR rank is set to zero). To verify information exists to calculate CR, a second ray cast from a sampled view pose to OI is performed within octree $\mathcal{M}$, and if the ray cast does not terminate on an occupied or frontier leaf, the CR rank is set to zero. These constraints quickly ignores view poses obstructed by obstacles or view poses that do not contribute information to calculate CR.

Our NBV algorithm terminates when all NBV CRs equal zero. This occurs when the number of frontiers is below a minimal viewing threshold with non-zero occupied leafs in the octree (i.e. η_{FRi}<thresh and $\eta_{OCi} \neq 0$).

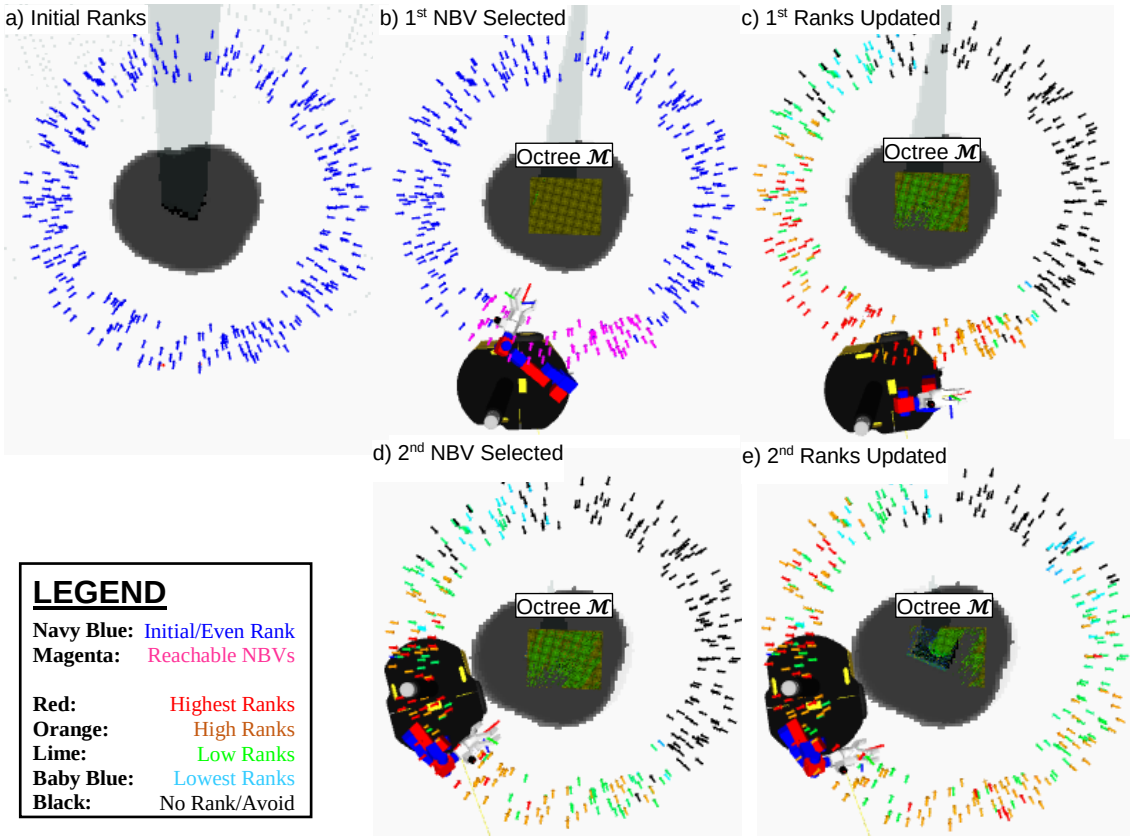


Figure 3.4: Next Best View (NBV) Ranking Progression

CR behaviour for NBVs is visualized in Figure 3.4. Initially, all NBVs are assigned equal ranks, shown as navy blue. The mobile manipulator moves to a location where a group of NBVs are reachable, shown in magenta, and randomly selects any available NBV to perform the first scan. Once octree $\mathcal{M}$ is update, CRs are updated to determine the next scanning location. The mobile manipulator travels to the highest CR, shown in red, and the process repeats until all CRs return back to zero rank. The number of zero ranks diminish as scans proceed because more frontiers are revealed. Eventually, as more frontiers are scanned and removed from octree $\mathcal{M}$, zero ranks reappear and eventually envelop the OI when fully scanned.

3.5. Registering Partial Scan with Uncertainty

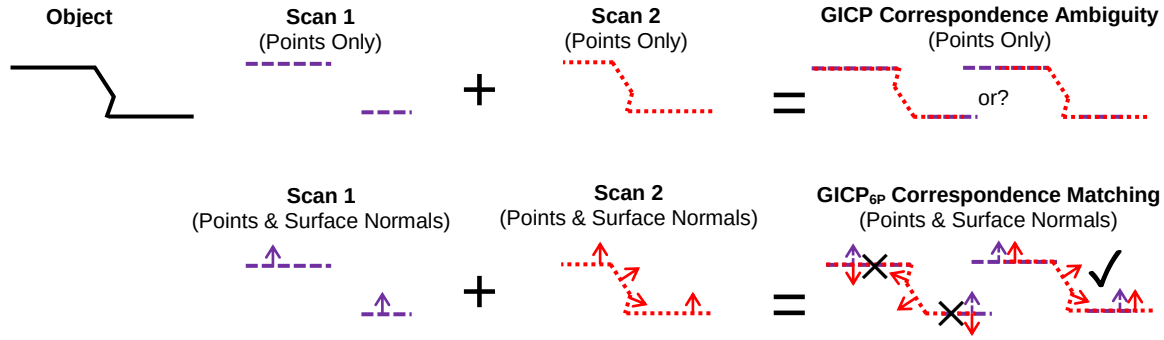


Figure 3.5: $GICP_{6P}$ Incorporates Surface Normals for Improved Correspondence Matching

Consecutive scans, *S*, are registered using GICP. This algorithm improves the original ICP's performance by associating a covariance to each registered point, and updating its covariance using a surface normal. GICP derives a transform that minimizes the distance between target and source points lying along a similar surface plane[66]. Implicitly, (G)ICP algorithms guarantee convergence when registered scans fully overlap; when scans partially overlap, convergence is not guaranteed because correspondence between target and source points might not be correctly identified. Correspondence between target and source points are determined using an L^2 norm which excludes points beyond a maximum distance, i.e. d_{max}, threshold[66]. A few issues with this approach exist: i) small overlap between scans, relative to d_{max}, causes poor registration because correspondence incorporates more non-overlapped points, and ii) if uncertainty exists, d_{max} needs to be increased to incorporate more non-overlapping points during registration.

Works generally add an additional stage to ICP to improve correspondence matching [67, 68]. A branch and bound algorithm minimizes residual error from the L^2 norm and guides ICP to a globally convergent solution[67]. A point's curvature and angle differences between neighbouring normals are estimated to discern pairs of points for registration[68]; in addition, the search space for correspondence is reduced through feature extraction[69]. Our GICP algorithm (i.e. $GICP_{6P}$) determines correspondence using the L^2 norm with a 6D feature created by concatenating a 3D point with its 3D normal. Since a surface normal can be resolved in two opposing directions, the direction facing the lidar's view is chosen. Scale between translation (i.e. points) and orientation (i.e. surface normal) can be adjusted using a scale factor. Applying the L^2 norm to a 6D feature vector improves correspondence because dissimilar normals increase the L^2

norm distance, guiding correspondence to select physically close points with similar normals. GICP's *O*(Nlog(N)) complexity remains unchanged. Alternatively, GICP's L^2 norm correspondence metric can be replaced with inner product to measure distance; an added scale factor would relatively weight point and normal distances. Empirically, we observed $GICP_{6P}$, using a scale factor equal to 1, registers points more accurately than standard GICP. Figure 3.5 illustrates how concatenating points (i.e. dotted lines) with surface normal information (i.e. arrows) improves correspondence matching. $GICP_{6P}$ ignores initial transformations that align dissimilar surface normals that are acceptable for GICP. Consequently, several local minima registration cases are avoided.

3.6. Correcting Grasp Pose Uncertainty (Pose Correction)

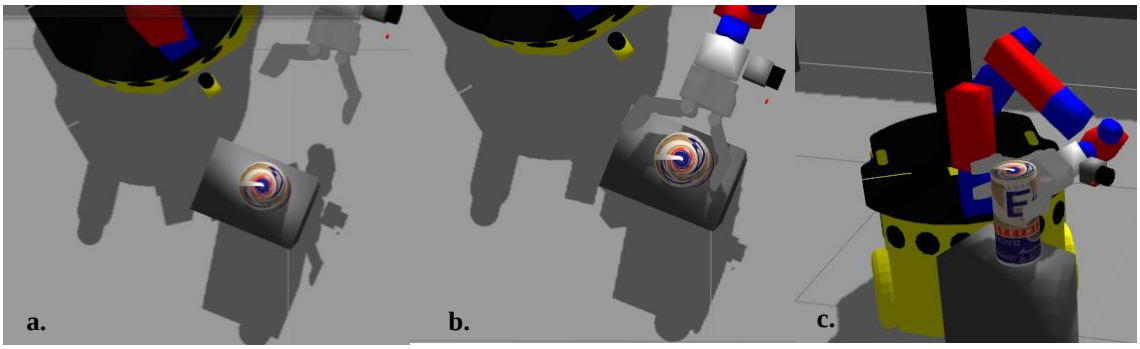


Figure 3.6: Grasp plan generated for the OI. The manipulator moves the gripper to its pre-grasp position (a) and a cartesian planner moves the gripper forward to reach its final grasp pose (b & c)

Given a set of potential grasp poses, random base positions around the OI are sampled with the robot facing the OI. For each base pose, the inverse kinematics (IK) are numerically calculated for pre-grasp and final grasp poses, $\mathbf{G}_p$ and $\mathbf{G}_f$, respectively; in addition, a straight line trajectory is processed from $\mathbf{G}_p$ and $\mathbf{G}_f$. If reaching a path to $\mathbf{G}_p$, $\mathbf{G}_f$, and all waypoints between $\mathbf{G}_p$ and $\mathbf{G}_f$ are successful, the base pose's rank is incremented. As shown in Figure 3.6 the manipulator executes a trajectory to $\mathbf{G}_p$ with an open gripper, and at $\mathbf{G}_p$, a Cartesian planner executes a straight forward trajectory to $\mathbf{G}_f$ and closes the gripper around the OI to trap it. When the robot reaches its base goal to grasp, its world frame location is uncertain; errors from odometry and mapping during the last base trajectory will alter the pose of the OI (relative to the base frame). To mitigate this error, the eye-in-hand sensor scans the OI again to correct the final base and grasp poses using the corrected transform derived by GICP.

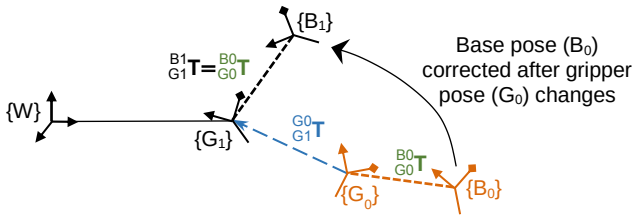


Figure 3.7: Transforms for Final Grasp and Base Pose Correction.
This figure is reprinted with permission from [52]

After every consecutive scan, GICP registers the OI's point cloud to the most current scan. Planned grasp goals can be registered with this transform because grasp poses are relative to the OI's surface features. Planning can derive a complete base and manipulator trajectory to execute a final grasp goal, but once the mobile base moves to a final base pose, uncertainty is reintroduced; the base will not reach its intended planned base goal. However, if the OI is scanned again, GICP's transform can correct the planned base goal. As a result, no manipulator trajectory replanning is necessary; the mobile manipulator moves a relatively short distance (i.e. uncertainty does not accumulate significantly) to the corrected base pose to execute its original manipulator trajectory to the final grasp goal.

As shown in Figure 3.7, $\mathbf{G_0}$ and $\mathbf{B_0}$ are planned poses for the gripper and base in the world frame (W) to pick the OI. GICP yields ${}^{G0}_{G1}\mathbf{T}$. The goal is to discover a corrected base goal, $\mathbf{B_1}$, to reuse the same planned manipulator joint configuration to reach the corrected gripper pose $\mathbf{G_1}$. $\mathbf{B_1}$ maintains the same pose relative to $\mathbf{G_1}$ using the constraint ${}^{B1}_{G1}\mathbf{T}={}^{B0}_{G0}\mathbf{T}$. The GICP transform within W with its rotation (zero translation) denoted as ${}^{G0}_{G1}\mathbf{T}_R$, where $\mathbf{T}\in\mathbb{R}^{4\times4}$. Pose of the $\mathbf{G_0}$ frame origin ${}^{W}_{G0}\mathbf{P}_{org}$ and rotation ${}^{W}_{G0}\mathbf{T}_R$ are corrected with the following:

$$ {}^{W}_{B1}\mathbf{P}_{org} = {}^{G0}_{G1}\mathbf{T}\,{}^{W}_{G0}\mathbf{P}_{org} - {}^{G0}_{G1}\mathbf{T}_R\,{}^{B0}_{G0}\mathbf{P}_{org} $$

$$ {}^{W}_{B1}\mathbf{T}_R = {}^{G0}_{G1}\mathbf{T}_R\,{}^{W}_{B0}\mathbf{T}_R $$

3.6.1. Reservoir Sampling to Mitigate Grasp Pose Replanning

Base pose ranks described above prioritize a base pose that successfully reaches the most final grasp poses. However, if experiencing large uncertainty, final grasp positions could be corrected and offset a large physical distance away from prediction. The

system shown in Figure 3.1 permits replanning (i.e. selecting a new base pose to regrasp), but planning, with uncertainty, can mitigate this action. The most significant source of error for grasping is position uncertainty because a poorly positioned gripper can bump or not even touch the OI to cause a failed grasp. When utilizing a trapping strategy (i.e. closing a gripper around an object), successful grasps are shown to still be possible with minor orientation error, i.e. less than 20°[70, 71].

To represent grasp position uncertainty, a box is defined around $\mathbf{G}_p$. The box's dimensions are defined to represent expected uncertainty. The pre-grasp pose, $\mathbf{G}_p$, is cloned to fill the box at discrete intervals. Reservoir sampling[72] removes a percentage of cloned poses. Base pose ranks are recalculated, as described in Section 3.6, for the remaining cloned $\mathbf{G}_p$ poses. This new rank prioritizes a base pose that can be offset to the most positions and can still successfully reach a final grasp goal.

3.7. Experiments and Results

3.7.1. Implementation

Framework implementation is developed within the Robot Operating System (ROS). Within ROS, MoveIt!'s interface[61] is configured to implement RRT-Connect from the Open Motion Planning Library[73] to guide our 6-DOF manipulator along collision-free trajectories [65]. The goal and orientation tolerances for the end effector are set to 0.5cm and 1° respectively. Manipulator joint tolerances are set to 0.28°. Base navigation, utilizing ROS' dynamic window approach planner[74], is performed with a base pose tolerance of 15cm and 5°. Base translational and rotational velocities are limited to 0.7m/s and 57.3°/s respectively. During grasp execution, base pose goal tolerance is reduced to 4cm and 5°; translation and rotational velocities are also reduced to 0.15m/s and 14.3°/s to prevent the base from oscillating at the base goal.

Two octomap servers are launched. World octree $\mathcal{W}$ is used for collision avoidance to safely move the manipulator. Octree $\mathcal{W}$ encompasses the room, has a resolution of 10cm and is updated by the Velodyne and Hokuyo sensors. Three hundred NBVs are randomly generated between a radius of 1.0m and 1.5m from the OI. This range is selected to permit the base to freely move around the OI. Octree $\mathcal{M}$ surrounds the unknown OI and updates NBV rankings to guide model reconstruction[57]. Object modelling is guided using a 70cm x 70cm x 50cm octree $\mathcal{M}$ surrounding the OI with its resolution set to 0.8cm. Any object(s) set within this bounding box is processed for modelling and grasping. A ray-cube intersection algorithm detects when the eye-in-hand laser data passes through octree $\mathcal{M}$'s bounding box, and if this occurs, laser data is virtually terminated along the bounding box's border[58]. Practically, this step is necessary to update the octomap because its current implementation only updates if a 'hit' occurs within its defined bounds. Unorganized point clouds model the object and are processed using the point cloud library (PCL)[75].

3.7.2. Object Reconstruction by Varying Overlap

To demonstrate our NBV algorithm, our system was first configured to autonomously scan the OI after selecting five NBVs with ω = 0.3 to ω = 0.7 at 0.1 increments. The scene and OI never changed during runs. A model of the OI was first constructed by

manually choosing 32 scans to create an approximate ground truth reference model (M_R) for validation. A model was created from real data because our simulation environment, i.e. ROS Gazebo, is intended to model physics and represents objects with meshes (without a ground truth point cloud). Initially, M_R is registered to the first scan S_0. During the experiment, GICP registers subsequent scans (i.e. $S_1, S_2, \ldots S_i$) to reconstruct the OI (M_i). Two variants of GICP are implemented with identical parameters, denoted as $GICP_{3P}$ and $GICP_{6P}$, when correspondence is estimated from 3D and 6D points respectively (described at the beginning of Section 3.5). Three transforms are recorded during each scan for comparison. The first two are: i) $GICP_{3P}$ transform ($\mathbf{T}_i^{3P}$) that registers current scan S_i with current model M_{i-1}, i.e. $S_i \cup M_{i-1}$, and ii) $GICP_{6P}$ transform ($\mathbf{T}_i^{6P}$) that also registers $S_i \cup M_{i-1}$. The third transform ($\mathbf{T}_i^{R}$) registers $S_i \cup M_R$ and represents the 'ideal' transform. All constructed models are down-sampled to 4mm resolution, and two metrics evaluate their construction:

Transform Similarity

Table 3.1: Comparison of $GICP_{3P}$ and $GICP_{6P}$ to Ground Truth.
This table is reprinted with permission from [52]

	Registration Failure (Scan #)		***Ave. Transform Similarity (cm)***	
ω	***GICP$_{3P}$***	***GICP$_{6P}$***	***TS^{3P}***	***TS^{6P}***
0.3	N/A	N/A	3.86	0.78
0.4	3	N/A	3.73	1.88
0.5	2	4	20.06	3.02
0.6	2	4	8.96	6.20
0.7	2	4	8.67	4.69

Transform similarity, TS_i^{nP}, is the difference in distance after applying the estimated transforms $\mathbf{T}_i^{nP}$ and $\mathbf{T}_i^{R}$ to a point, $\mathbf{P}_{org}$, located at the origin of the world frame. At the end of five scans (i.e. k=5), the results are averaged.

$$TS_i^{nP} = \frac{1}{k}\sum_{i=1}^{k} \left\| \mathbf{T}_i^{nP}\mathbf{P}_{org} - \mathbf{T}_i^{R}\mathbf{P}_{org} \right\|_{n=3,6}$$

Registration Failure

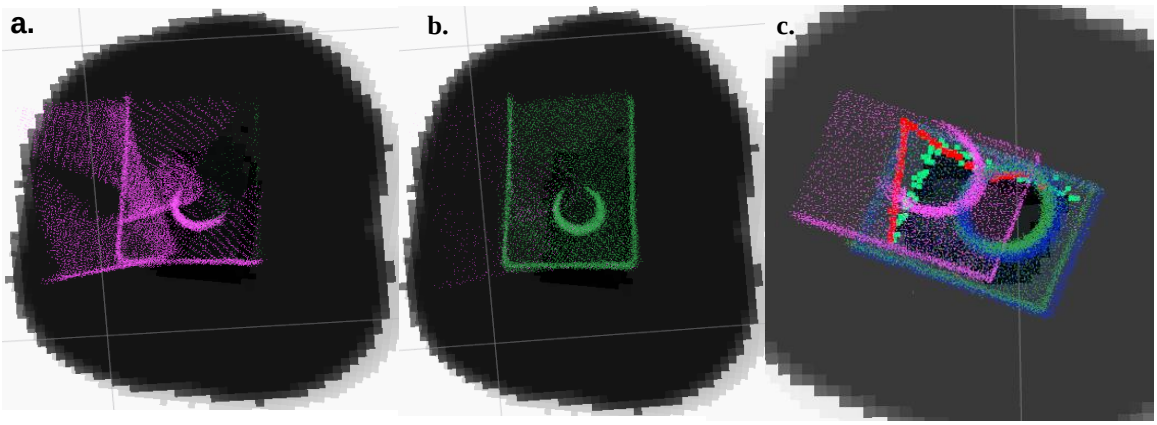


Figure 3.8: $GICP_{3P}$ (left) & $GICP_{6P}$ (middle) merged scans.
Offline model (blue) and $GICP_{3P}$ (pink) / $GICP_{6P}$ (green) point clouds overlayed (right).
This figure is reprinted with permission from [52]

Every consecutive scan will accumulate a small error over time. Empirically, we notice registration typically fails when $TS_i^{nP} > 5cm$, and we record it when this occurs. Table 3.1 summarizes registration results by modifying the overlap factor ω. As ω increases (i.e. reducing desired overlap), average transform similarity decreases but not monotonically. Poor TS^{nP}, shown when ω = 0.5, is due to $GICP_{3P}$ registering a scan to create a model that looks like Figure 3.8a, instead of being aligned like Figure 3.8b. Figure 3.8c demonstrates the difference between magenta $GICP_{3P}$ and green $GICP_{6P}$ models registering to the ground truth (shown in blue). This is caused by several factors, like little correspondence or discernable features captured between scans. Regardless, poor registration causes a failure to determine a valid grasp. By selecting an appropriate overlap factor ω, registration failure can be mitigated as shown in Figure 3.8b.

Given sufficient overlap, registrations estimated from $GICP_{3P}$ and $GICP_{6P}$ are similar. Referring to Table 3.1, $GICP_{3P}$ and $GICP_{6P}$'s desired overlap can be selected as ω = 0.3 and ω = 0.4 respectively to avoid registration failure. Average TS^{6P} is consistently less than TS^{3P}; $GICP_{6P}$ transforms are more similar to our approximate ground truth model. Consistently, $GICP_{6P}$ registers scans more accurately (shown in Figure 3.8b and Table 3.1) and registration failures occur less frequently when concurrent scans have more than 50% overlap, ω < 0.5. This can be viewed in Figure 3.8c as poor registration causes the $GICP_{3P}$ model (shown pink) to translate ~7cm away from the ground truth (shown blue).

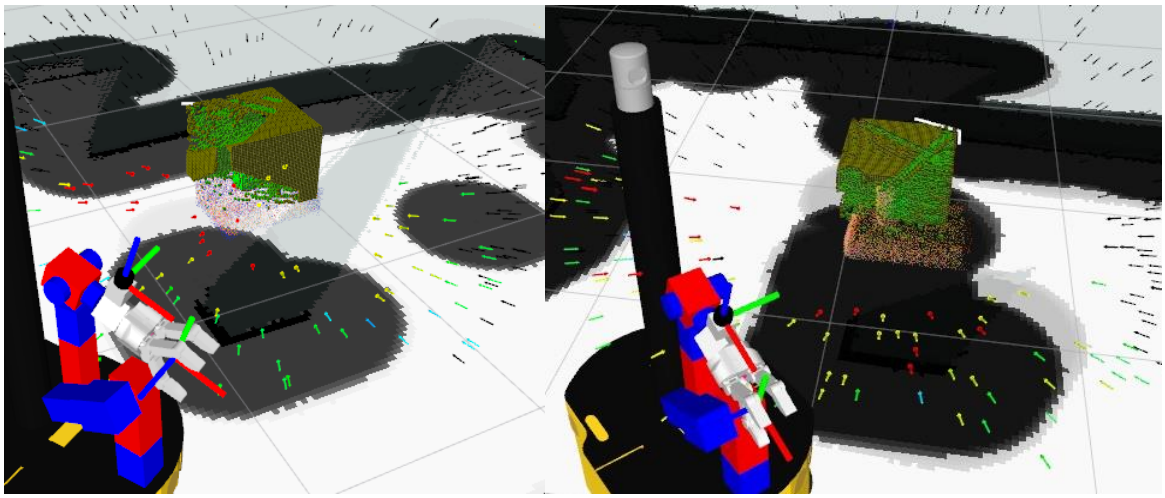


Figure 3.9: NBV Ranks Before and After a Scan to Reveal Cylindrical Object (ω = 0.5)
(Warmer coloured arrows have the highest ranks).
This figure is reprinted with permission from [52]

Figure 3.9 visualizes NBV rankings when ω = 0.5 (i.e. 50% overlap is desired). In the left image, the eye-in-hand laser scanned the object's bottom and table surface. The point cloud, shown as copper coloured points, is observed below octree $\mathcal{M}$, shown as a green voxelized box. Highest NBV ranks (shown as red arrows) orient towards the top of the object because these orientations overlap with the previous scan. After selecting the highest ranking NBV (i.e. scanning the object's top), the right image shows how the ranks change; red arrows migrate outwards because the object's exposed sides have scan overlap with the previous two scans, and more points from the copper coloured point cloud are revealed. This behaviour repeats until the object is fully modelled. When ω < 0.5, NBVs are ranked to orient closer to the previous scan. When ω < 0.3 or ω > 0.7, ranking behaviour from the NBVs is not discernable; when less than 0.3, ranks prioritize viewpoints towards the last scan, and above 0.7, ranks prioritize viewpoints towards the opposite side of the OI. Warmer coloured arrows indicate highest NBV ranks and black arrows are ignored NBVs. Initially, all NBVs start and finish with equal ranks, indicated by navy blue.

3.7.3. Complete Object Reconstruction

In simulation, a tin can is reconstructed until our NBV modelling algorithm self-terminates (i.e. all CRs are zero), using a scan overlap setting ω = 0.4. At each scan location, NBV ranks, travel distance, octree $\mathcal{M}$, and underlying point cloud model are shown in Figure 3.10 and Figure 3.12. Scanning begins at Figure 3.10b and Figure 3.12a where the

mobile robot randomly selects a reachable NBV to begin object modelling, and subsequent images depict the mobile manipulator scanning at the next NBV. The tin can model and reconstructed point cloud model are observed in Figure 3.11. A more complex scene including a tin can and cordless drill model reconstruction is observed in Figure 3.13.

Figure 3.10 and Figure 3.12 show NBV ranking progression as the OI is revealed and completely scanned. They also show base pose locations the system selects for scanning. Black arrows indicate zero rank NBVs (i.e. view poses to ignore), described in Section 3.4.2, because either not enough information exists to calculate a CR or the object is occluded by an environmental obstacle. Warm colours, i.e. red arrows, indicate NBV poses prioritized for scanning. Generally, after the first scan, warm NBVs prioritize poses neighbouring the previous scan. Due to the nature of the frontier leaf's definition (i.e. a free leaf next to an unknown), few frontiers are discovered initially. As a result, many NBVs that surround octree $\mathcal{M}$ are ignored (and shown as black arrows), except for view poses near the initial scan.

Iteratively, the mobile manipulator moves to neighbouring view poses and builds a complete object model, shown in Figure 3.11 and Figure 3.13. The system does not oscillate (i.e. make large movements) back and forth while scanning. As more scans continue, zero ranked NBVs disappear because occupied and frontier leafs are observable at most view poses. As more scans continue, frontier leafs are removed because more unknown leaf locations are scanned. During this phase, zero ranked NBVs reappear because all necessary information is gained (or frontiers cannot be observed) at these view poses. Continued scans remove more frontiers from octree $\mathcal{M}$, creating more zero ranked NBVs, until octree $\mathcal{M}$ is only comprised of free, occupied, and unknown leafs. Unknown leafs exist, but they do not contribute to generating a frontier because they are encapsulated by occupied leafs. When all frontiers are removed, all NBVs will become zero ranked, and the algorithm self-terminates.

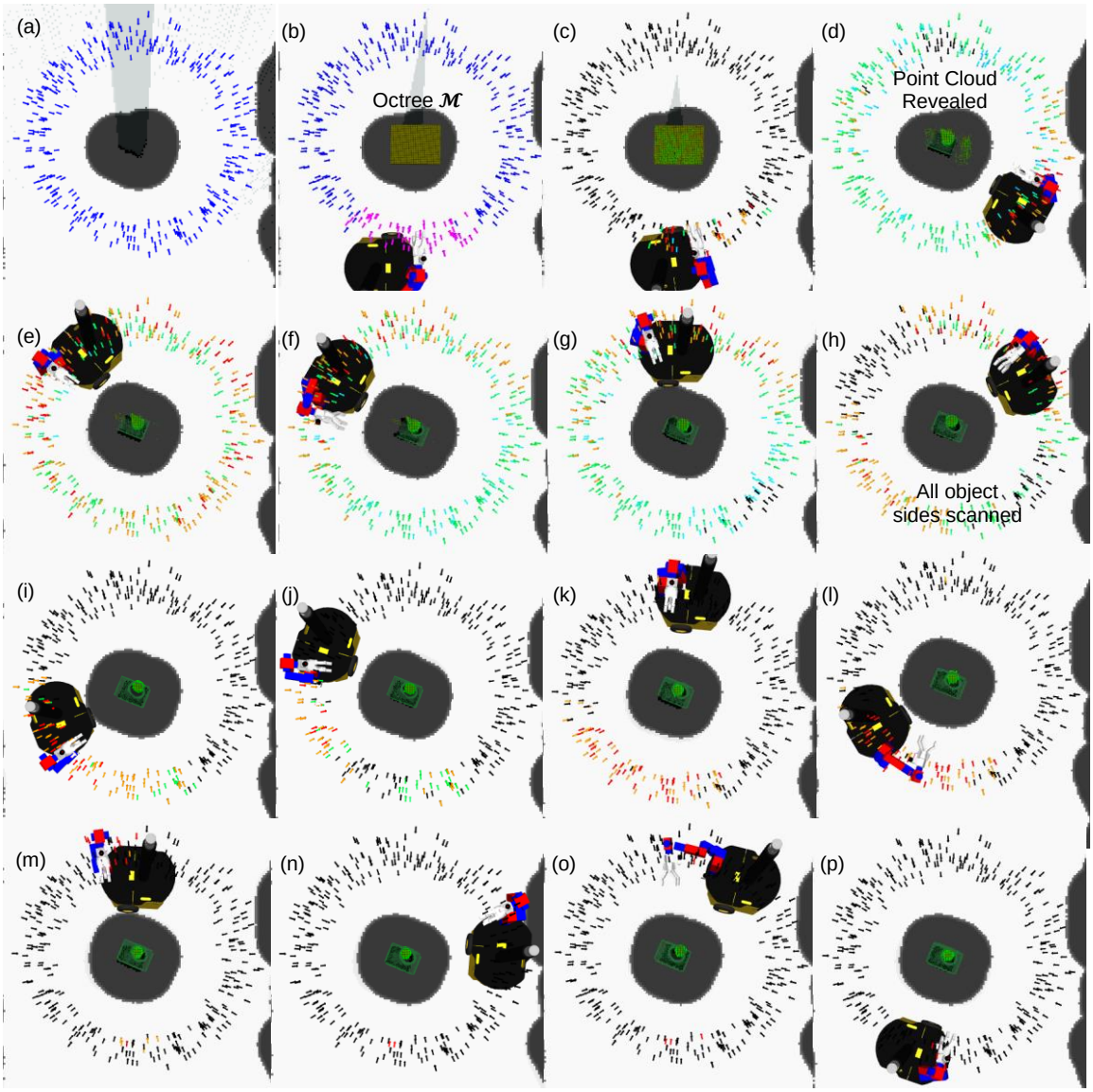


Figure 3.10: Mobile Robot Sequence Reconstructing a Tin Can Model, ω = 0.4
(Warmer coloured arrows have the highest ranks)

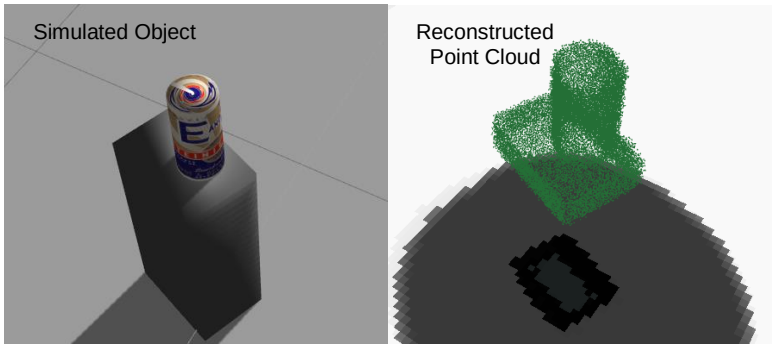


Figure 3.11: Tin Can Model Reconstruction Result, ω = 0.4

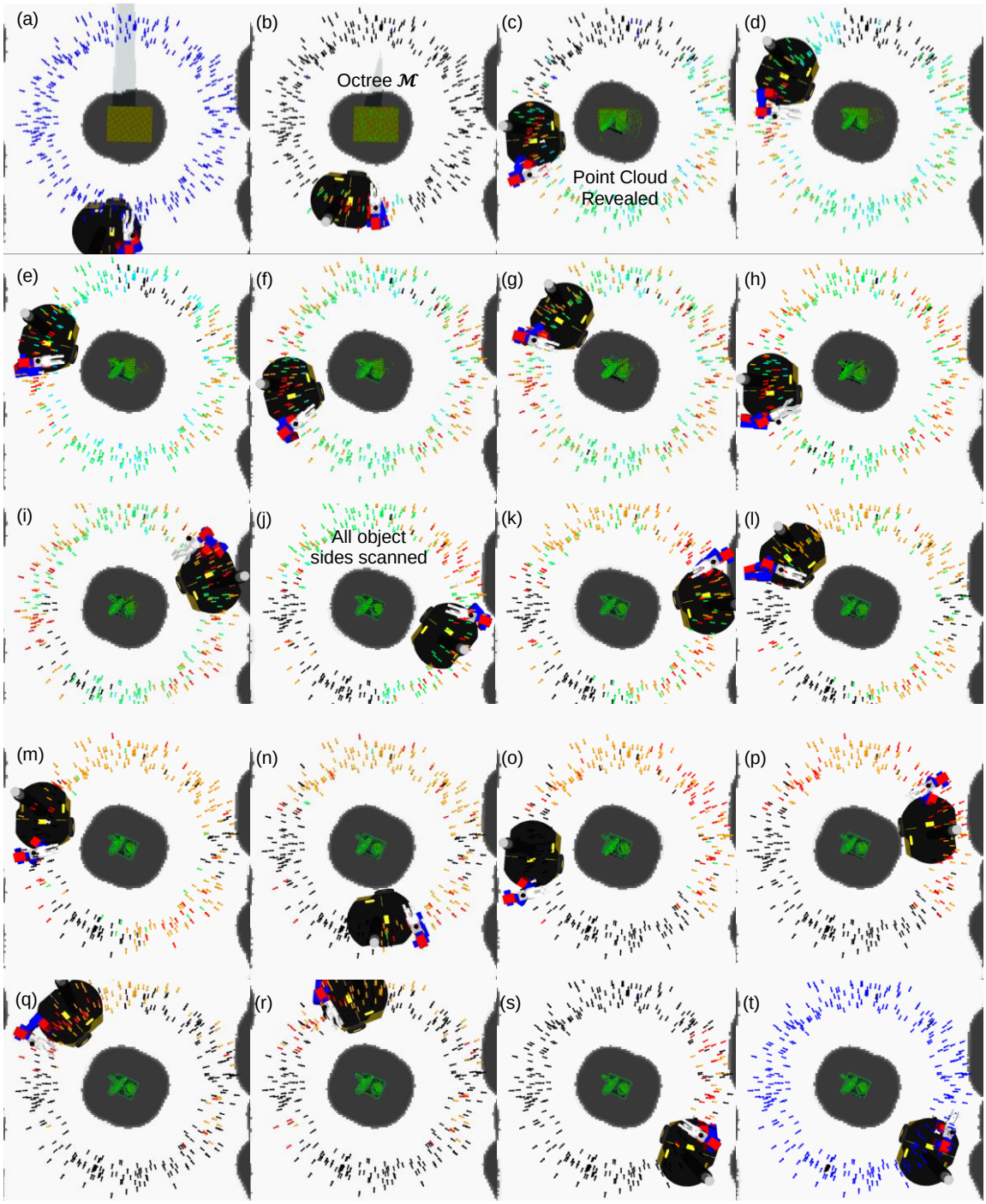


Figure 3.12: Mobile Robot Sequence Reconstructing Tin Can and Cordless Drill Models, ω = 0.4
(Warmer coloured arrows have the highest ranks)

In both simulation, our modelling algorithm can reconstruct point cloud models and self-terminate. The underlying point cloud models are initially observed after the third scan, i.e. Figure 3.10d and Figure 3.12c. All object sides are scanned after eleven NBV poses are selected, i.e. i.e. Figure 3.10h and Figure 3.12j. Modelling the tin can and cordless

drill required five additional scans compared to modelling the tin can only; this is expected because the cordless drill's irregular shape self-occludes itself and the other object. Both reconstructed point clouds, Figure 3.11 and Figure 3.13, resemble their respective virtual models and show minor registration error.

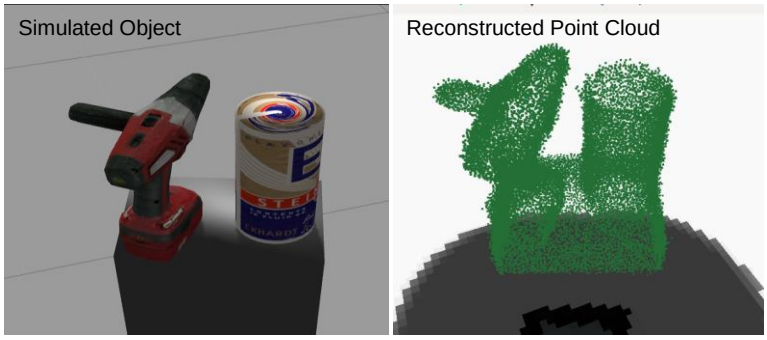


Figure 3.13: Tin Can and Cordless Drill Model Reconstruction Result, ω = 0.4

3.7.4. Framework Performance

To demonstrate performance, three live experiments are performed to grasp the OI, a can, while reconstructing its partial point cloud model. The OI's surface is covered with construction paper to create a Lambertian surface to improve the Hokuyo laser's sensor data quality. NBVs are ranked with ω = 0.40, and grasp successes (or failures) are recorded.

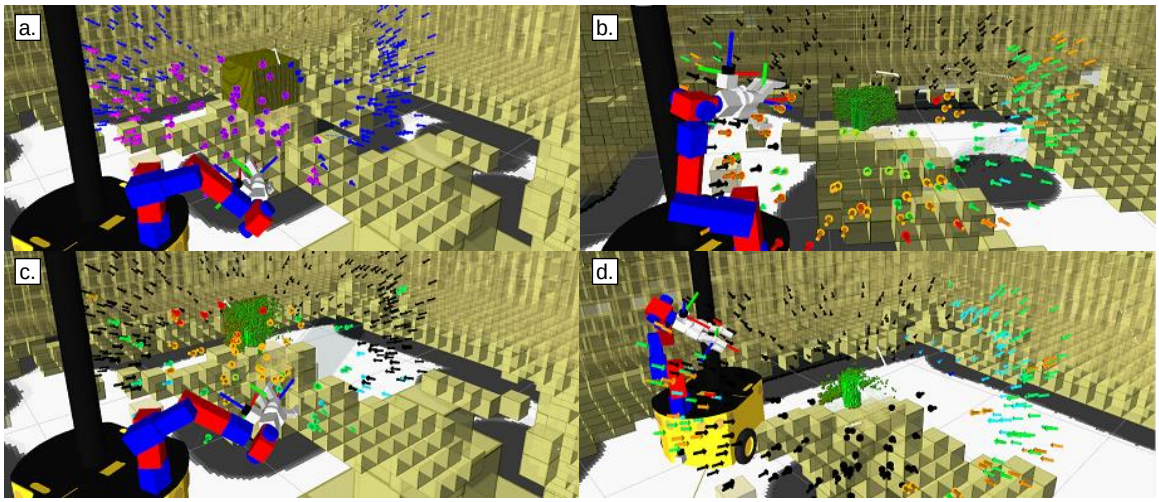


Figure 3.14: Object Modelling Progress while Avoiding Obstacles.
This figure is reprinted with permission from [52]

To fully model an object with ω = 0.4, the robot needs to move between seven to twelve view poses. Within five view poses (i.e. within fifteen minutes), the robot scans enough

of the object to discover a grasp pose. The total time planning a trajectory to an NBV ranges from 45s to 90s; planning a grasp ranges from 45s to 400s. Grasp planning time increases as more grasp goals are searched or if the base corrects and replans its final position. Grasp analysis completes within 45s. Figure 3.14 shows the progression to model an object while avoiding environmental obstacles. These images show the base is positioned to keep the manipulator free to scan the OI. By the sixth scan (i.e. Figure 3.14d), most of the object is revealed. Due to the overlap constraint specified for the NBV ranks, scans merged well to perform grasp analysis.

Work presented in [76] suggests that an object's principal axis can be used as a grasp feature. In fact, [18] completes all grasps based only from a center of mass (CoM) and principal axis estimates from a partial point cloud. We have encountered two issues using the principal axis to guide a gripper's pose: i) it works well for long, symmetric objects but produces incorrect results for objects with an irregular shape (e.g. a cordless drill) and ii) the principal axis does not stabilize when the OI is partially modelled. Orientation tends to follow along the point cloud direction that contains the highest density of points instead of the OI's true pose, and the priciple axis tends to bias towards the scanned sides of incomplete model. As our modelling phase guarantees overlap but not uniform sampling, we avoided estimating the object's principle axis.

Table 3.2: Framework Performance Summary.
This table is reprinted with permission from [52]

#	*Model Locations*	*Base Pose Corrected?*	*Grasp Pose Corrected?*	*L^2 error to Goal (uncorrected)*	*L^2 error to Goal (corrected)*
1	2	No	Yes	6.40cm	0.01cm
2	2	Yes	Yes	4.05cm	0.06cm
3	1	No	Yes	7.90cm	<0.00cm

Table 3.2 summarizes results to autonomously grasp an OI. Desired grasp poses are corrected, and the gripper reached its desired grasping goal. During the second run, corrected grasp poses fell outside the robot's dextrous workspace, and the base repositioned its location to complete the final grasp. Although not recorded, if the base pose correction is less than the goal tolerance of the base planner (<4cm radius), the base will not correct its desired base goal because it is already within its base goal threshold. While testing, more than one attempt at base correction has never been observed.

Chapter 4.

Generalizing Grasping for Multiple Grasp Types

4.1. Problem Statement

We present a generalized grasping approach for mechanical grippers that permits grippers with any number of fingers to discover poses for different grasp types. Grasp type matching is performed in a computationally efficient two stage process. In the first stage, a set of grasp type orientations that yield a “pure shape” match are discovered by matching histograms between finger contacts’ (corresponding to a grasp type) and objects’ surface normals. A pure shape match is when a gripper’s finger contact normal distribution matches that observed among a set of object surface normals. In the second stage, “size” of grasp type is matched by cross-correlating voxel grids representing the gripper with partially viewed objects. Furthermore, collision constraints, e.g. the gripper palm, can be accommodated in the second stage as one single step by introducing negative penalties during cross-correlation. Due to decoupling matching shape from scale, a grasp contact verification stage is added to verify grasp positioning after cross-correlation; finger contacts may physically match an object surface but gripper contact normals may not perfectly align with observed surface normals. A tolerance rejects poses when the angle between any gripper contact and object surface normal becomes too large.

Objects are assumed to be rigid, at rest, on a table, and their context is unknown. The presented algorithm should generate grasp poses along an object surface that match a grasp type to reasonably lift an object. Contextual information (i.e. objects are stacked, are slippery, fused to the table, need to be retrieved in a specific order) is not considered. Task information, (i.e. human-to-robot transfer or grasping securely) is decoupled from pose generation and incorporated as a final stage to refine pose generation results.

4.2. Introduction

Grasp planning and manipulation is fundamental in a variety of robotic domains, in particular for assisted robotics to aid people to complete a wide variety of tasks. In

robotics literature, grasps are planned based on analytical and heuristic approaches to securely grasp an object[8, 9]. For instance, force closure and caging works focus on analytical methods that rely on an object's geometry, kinematic, and/or dynamic equilibrium[77-80]. A limitation for these methods is their design is typically for one specific task wrench (such as resisting gravity) or grasp type (such as a two fingered pinch grasp). As pointed out in [43], newer learning-based methods tend yield a single grasp type per object for multi-finger grasps.

4.3. Motivation

Planning for a single grasp type is rather limiting because the grasp type itself is determined from the task being accomplished, e.g. a power grasp to hold a tool, tripodal to grasp a ball, or precision to hold a pen[43]. Determining a grasp itself is not simply securing an object with parallel jaws, rather it is completing a practical task comprising of multiple sub-tasks which may require different grasp types. Securing an object is only one sub-task; other task types include object transfer, object manipulation, tool usage, etc. The principle objective of our research is to simultaneously generate grasp poses for different grasp types to accomplish a task.

A key feature of our work is discovering grasp candidates for multiple grasp types without explicitly estimating an object's grasp stability. A stability check is forgone because our approach is designed to work with partial object information; such a stability check would generally require complete object information. When quantifying grasp robustness, a contact region (and not a point) can potentially provide stable grasps over large perturbation between an object's surface normal and grasp axis angle[81]. Capturing this behaviour is desired for systems experiencing uncertainty. We posit that a grasp taxonomy's grasp type[82, 83] is inherently stable if defined correctly, and our algorithm is motivated by this observation. For example, a pinch (or parallel) grasp applied by a human to a plate's corner may not be stable due to torque allowing the plate to rotate between each finger; however, this is a common grasp for humans (who have soft fingers[84]) because more force can be applied at one's fingertips to increase friction and prevent rotation. Many newer multi-fingered grippers (e.g. Barrett, Schunk, and Robotiq) have soft material enveloping their fingers that could resist torque similarly. Discovering grasp candidates for a grasp type is not 'optimized'; we argue optimizing (or selecting) a grasp should be a final decision because any optimization is dictated by a

task—e.g., if a task is securing an object, force closure and caging are optimized to restrict an object's dynamic motion, but the consequence is the gripper contacts will ideally surround an object. If the task changes to handover, a sub-optimal grasp (in terms of minimizing dynamic motion) is desired to reveal more surface area for the receiving person to grasp.

In grasping literature, there is an implicit theme to place importance on object shape. Object shape requires an appropriate grasp taxonomy[82], but this leads to a significant problem; an infinite number of shapes requires numerous grasp types. A human hand is observed to have over thirty different grasp variations[82, 83, 85]. Computing all grasp types for all objects would be an endless task. However, the search space for grasping can be reduced if we focus to generalize grasp types rather than an object's shape. The key idea in our approach is to discover if a grasp type exists or "matches" to a set of contacts on an object's surface; we assume a grasp is achievable if it exists. If a grasp type does not exist, that grasp cannot be executed. Even though over thirty grasp types exist for a human hand, these types can be generalized to a smaller set for a mechanical gripper to complete a task. (i.e. pinching, power, tripodal, parallel, ring, etc.)[85].

4.4. System Overview

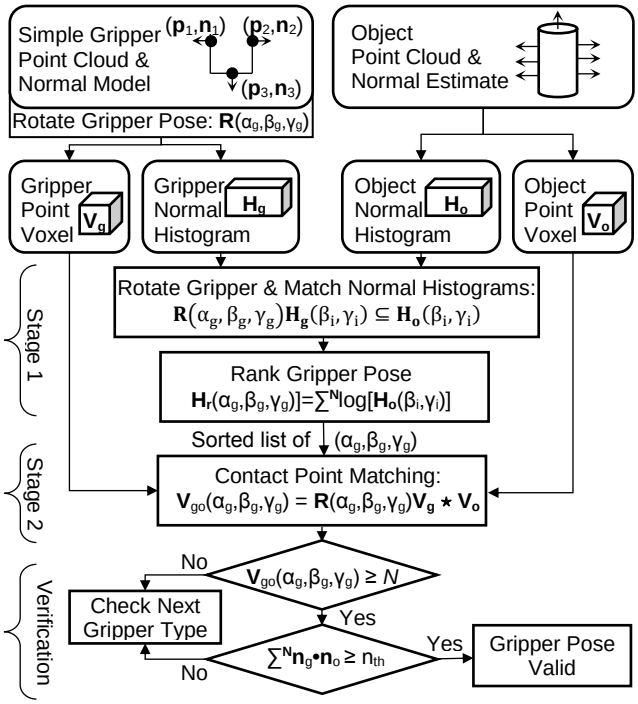


Figure 4.1: Grasp Pose Pipeline Overview.

Our system determines a grasp pose by cross-correlating a gripper's grasp type shape with an object's surface. Grasp type shape is modelled by desired contact surface normals. For example, a parallel jaw can be represented by either two or an array of opposing contact normals. Grasp types selected for multi-fingered grippers is inspired by human grasp taxonomy, and we investigate modelling power, tripodal, and lateral grasp types[83]. We assume object(s) being modelled are represented by a point cloud, a collection of points located on an object's surface (obtained via a range sensor) and their respective normal (estimated from the point cloud). Our algorithm's flow diagram is shown in Figure 4.1. A grasp plan for a single grasp type is completed in two stages using two data structures: 1) surface normal histograms, denoted by $\mathbf{H_o}$ for object surface normals and $\mathbf{H_g}$ for finger or grasp contact normals, and 2) 3D voxel grids, $\mathbf{V_o}$ for object voxel grid and $\mathbf{V_g}$ for gripper voxel grid, computed from point clouds. Furthermore, by adding negative penalties to the gripper voxel grid, a collision-free grasp and a "straight line path" to the object can be predicted in one integrated step. A surface normal histogram discretizes surface normals to their respective spherical coordinate angles, inserts them into bins, and shows the surface normal frequency for an object. Grasp contact normals for any grasp type are used to create $\mathbf{H_g}$ and are defined a priori. The advantage to using surface normal histograms is that they are invariant to translation and rotation from any viewing angle within the world frame. High frequency noise is filtered because quantizing surface normals smooth these frequencies.

Computational efficiency for our system is achieved by decoupling shape matching (using contact surface normals) from scale (using contact points). A similar matching result may be achieved by cross-correlating gripper contact normals and points in 6-dimensions (6D). However, the computation time to cross-correlate hypercubes in the frequency domain is $O(v^D \log(v))$, where 'v' is the number of voxels needed to create a cube's length, 'D' is the spatial dimension, and v^D is the total number of voxels in the hypercube [86]; a decoupled approach in 3D results in significant computational efficiency.

4.5. Grasp Planning Overview

4.5.1. Stage 1: Match Grasp Type Shape

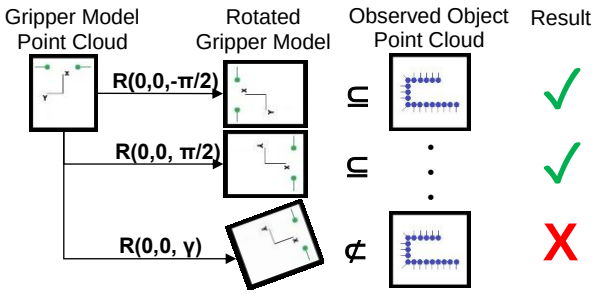


Figure 4.2: Stage 1 Example for Parallel Jaw to discover Gripper Orientation.

Stage 1 determines if a grasp type shape (not the scale) “matches” any part of viewed unknown objects, i.e. a grasp type’s set of contact normals are observed on any scanned object. A grasp type model is created using point normals to define desire contact points on a gripper. A straightforward measure for “matching” shape is pairing object surface normal bins to non-zero bin locations within the grasp histogram. A grasp type’s contact normals used to create $\mathbf{H_g}$, defined a priori, are rotated by the gripper’s orientation using Euler angles. For each gripper rotation, a rank quantifies how many object surface normals match desired gripper contacts. Heuristically, this step selects and ranks grasp orientations (i.e. top-down, sideways, etc.) that match the grasp type’s shape to any observed objects.

A Stage 1 example, shown in Figure 4.2, illustrates a parallel jaw gripper with two contact points matching a partially observed box (in 2D). Two point normals represent the gripper’s desired shape to contact the object. Constraints, i.e. a palm, are not illustrated because they are a physical shape constraint (i.e. not represented by point normals). In this example, the gripper model contact normals (shown as lines) align with the observed object normals only when the gripper pose is rotated ±π/2; thereby, rotating the finger contact normal by the same amount. Green lines are inverted gripper contact normals. Blue lines are outer surface normals associated to the object. These orientations will need further investigation to determine if the shape’s scale matches the gripper in a subsequent stage.

4.5.2. Stage 2: Match Grasp Type Scale

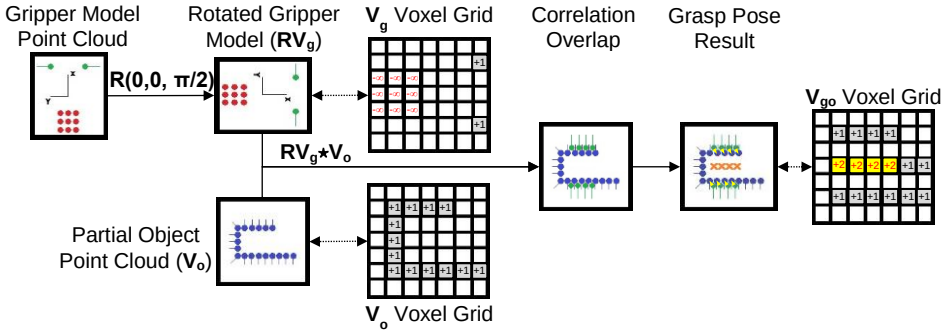


Figure 4.3: Stage 2 Example for Parallel Jaw to Discover Gripper Orientation.

Stage 2 determines if a grasp type's contacts points "match" a set of observed surface points on the object. Gripper orientations that satisfy Stage 1 are used to rotate a grasp type's point cloud model. Rotated points are then inserted into a gripper voxel grid ($\mathbf{V_g}$) and cross-correlated with the object's voxel grid (i.e. $\mathbf{V_{go}}=\mathbf{RV_g}\star\mathbf{V_o}$), where $\star$ denotes a cross-correlation operation and R is gripper orientation. Peaks from correlation greater than or equal to the number of grasp type contacts identify potential grasp locations.

Stage 2 for the parallel jaw gripper is shown in Figure 4.3, where parallel jaw gripper's green contact points are cross-correlated to an object's blue surface points; from cross-correlation, an orange 'X' identifies potential positions where the parallel jaw matches the object's surface, i.e. correlation is high. Symbolic red points are constraints applied to a region (with negative values) that should not collide with the object. Red points physically represent the gripper's palm and wrist. Lastly, a final step revisits potential grasp locations, finds the gripper contact points closest to a surface point in the object point cloud, and verifies each contact normal and object surface normal are in a "similar" direction. If similar, the grasp type's complete pose is inserted into a list for execution.

4.6. Grasp Planning Details

4.6.1. Stage 1a: Surface Normal Histograms

A surface normal histogram, shown in Figure 4.4, is created by representing a point normal, $\mathbf{n}=(n_x,n_y,n_z)^T$ via two spherical angles: 1) elevation / pitch β_i:$[-\pi/2,\pi/2]$ and 2) azimuth / yaw γ_i:$[0,2\pi)$, where 'i' is a coordinate (β_i, γ_i) within histogram $\mathbf{H}$.

$$\beta_i = \operatorname{acos}(n_z),\ \gamma_i = \operatorname{atan2}(n_x,n_y),\ n_z \neq \pm 1$$

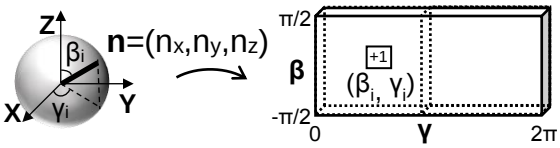


Figure 4.4: Surface Normal Histogram Data Structure.

Angles are discretized into uniform bins and incremented. Due to gimble lock when a normal is orientated at north or south poles, a unique solution to γ does not exist. For these two cases, all azimuth bins for the polar elevation angle are incremented $\Delta\gamma/2\pi$, where $\Delta\gamma$ denotes bin size. Normals for the gripper represent the grasp type's contact orientation; their orientation faces opposite to an object's surface normals so an inverted normal is inserted into $\mathbf{H_g}$, i.e. $\mathbf{H_g}[-\mathbf{n}]$.

4.6.2. Stage 1b: Matching Histograms

All histograms share the same angle resolution. If all non-zero gripper bin locations $\mathbf{H_g}(\beta_i, \gamma_i)$ are also non-zero at the same object histogram bin location $\mathbf{H_o}(\beta_i, \gamma_i)$, there exists a possibility the grasp shape is on the object's surface. However, this is only true for one gripper orientation. To match a grasp type shape for all orientations, *N* gripper contact normals $\mathbf{n} \in \mathbb{R}^{3\times1}$ are rotated using Euler angles roll α_g:[0,2π), pitch β_g:[-π/2,π/2], and yaw γ_g:[0,2π). A rotation transform is defined as $\mathbf{R_{zyx}}(\gamma_g, \beta_g, \alpha_g) \in \mathbb{R}^{3\times3}$. Rotated normals are then mapped to a histogram index, and these indexes are referenced to the object histogram to determine a rank. This method is fast because few gripper contacts are needed to represent grasp type. Note that gripper rotations correspond to bins shifting in $\mathbf{H_g}$.

4.6.3. Stage 1c: Ranking Different Orientations

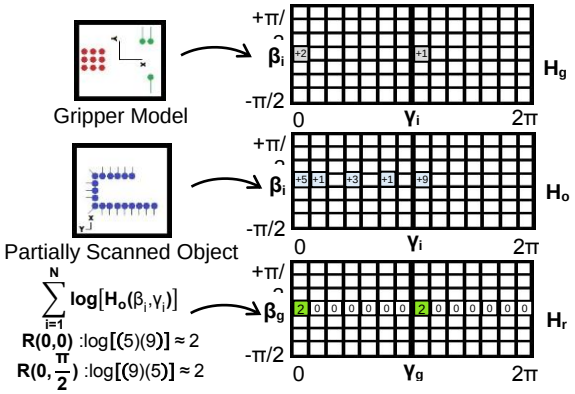


Figure 4.5: Surface Normal Histogram Matching (Bin Resolution = π/7).

Gripper orientation ranks are stored in a third surface normal histogram structure, i.e. a histogram result $\mathbf{H_r}$, whose structure is shown in Figure 4.4. For simplicity of visualization, the axis corresponding to roll α_g is not shown, but it is added to $\mathbf{H_r}$ as a third axis to index ranks for all possible rotations $\mathbf{R}(\alpha_g, \beta_g, \gamma_g)$. A bin in $\mathbf{H_r}(\alpha_g, \beta_g, \gamma_g)$ indexes a specific gripper orientation, and the rank value stored quantifies how well $\mathbf{H_g}$ matches with $\mathbf{H_o}$. Ranks for an *N*-contact grasp type are defined as:

$$\mathbf{H_r}\left(\alpha_g,\beta_g,\gamma_g\right) = \begin{cases} \sum_{i=1}^{N} \mathbf{log}\left[\mathbf{H_o}(\beta_i,\gamma_i)\right], & \mathbf{H_o} \geq \mathbf{R}(\alpha_g, \beta_g, \gamma_g)\mathbf{H_g} \geq 1 \\ 0, \text{ otherwise} \end{cases}$$

Rank is a rough measure to indicate the combination of each finger contact normal choosing a surface normal. If all normals match, the logarithmic product is performed at all $\mathbf{H_o}$ bin locations that correspond to non-zero $\mathbf{H_g}$ bin locations. Surface normals do not match when $\mathbf{H_g}(\beta_i, \gamma_i) \geq 1$ and $\mathbf{H_g}(\beta_i,\gamma_i) > \mathbf{H_o}(\beta_i,\gamma_i)$. This condition's intuition is too few object surface normals exist for the quantity of gripper contact normals requested. For this condition, $\mathbf{H_r}(\alpha_g, \beta_g, \gamma_g)$ is set to zero. Practically, thousands of surface normals can represent an object, creating large values within $\mathbf{H_o}$. Rank values can become extremely large using high resolution models. To mitigate this problem and keep rank values small, a log transformation is performed. Intuitively, rank maximizes when the most surface normals exist for each grasp.

Figure 4.5 illustrates a 3D example to create $\mathbf{H_r}(\beta_g, \gamma_g)$ using 2D rotation $\mathbf{R}(\beta_g, \gamma_g)$. All histograms have a resolution set to $\Delta\gamma = \pi/7$ (~25.7°). In this example, only yaw $\mathbf{R}(0, \gamma_g)$ is possible to rotate the gripper. If the gripper elevation angle changes, the same $\mathbf{H_r}$ results would shift up/down along β-axis for $\mathbf{H_r}$. Figure 4.5 demonstrates the logarithmic product rank. By comparing histograms, bins in $\mathbf{H_g}$ align with $\mathbf{H_o}$ when $\gamma_g = \{0, \pi\}$. For all other rotations, $\mathbf{H_g}$ bins do not match $\mathbf{H_o}$ and $\mathbf{H_r}(0,\gamma_g) = 0$. Although Stage 1 determines if a gripper orientation matches the grasp type shape to object surface, Stage 2 determines both the scale of the object shape and if it satisfies the gripper's physical constraints.

4.6.4. Stage 2a: Voxel Grid to Match Contacts

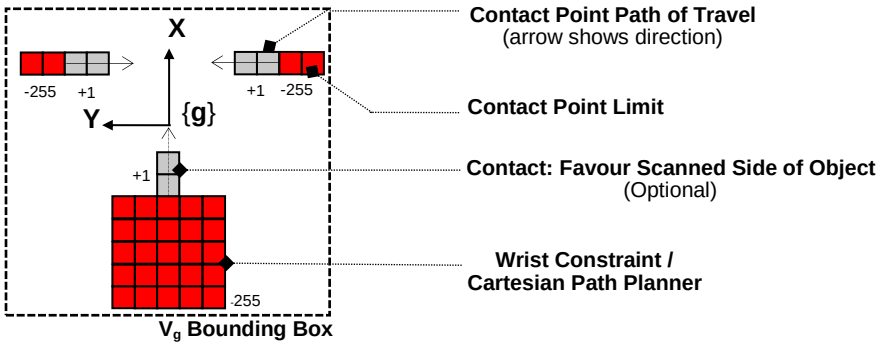


Figure 4.6: Voxel Grid Model for a 2-Contact Parallel Grip.

An object bounding box, $\mathbf{V_o} \in \mathbb{R}^3$, surrounds all object(s) being scanned, and a regular grid discretizes space into voxels. Each voxel is addressed by indexes, $\mathbf{I} = (i,j,k)^T$, where $\mathbf{V_o}(i,j,k) = 1$ if it corresponds to a scanned location and $\mathbf{V_o}(i,j,k) = 0$ if space is empty or unscanned (because unscanned regions also include a point cloud's subsurface that we want free). Discussed in this section's third paragraph, an additional palm contact added to a grasp model prevents poses to generate in unscanned regions. Shown in Figure 4.6, a gripper bounding box, $\mathbf{V_g} \in \mathbb{R}^3$, surrounds the gripper's grasp type being modelled and discretizes space with the same resolution as $\mathbf{V_o}$. $\mathbf{V_g}(i,j,k) = 1$ to model a contact point and its range of motion prior to contacting an object. This motion is defined by a directional vector for our system with adjustable length and represents a finger contact's motion just prior to contact. This generalizes a grasp type for different object sizes. Precise motion depends on the finger joint's kinematics, which can be modelled precisely. However, modelling straight line motion, i.e. a sliding joint, is computationally more efficient and a reasonable approximation for a relatively small range of motion; hence, we implemented this approximation for our system. We modelled rotational motion as a trial, e.g. revolute finger joints, but in practice, no significant effect was observed modelling an arc for rotational motion. Due to voxelization, an arc or chord passes through similar voxels over short distances.

In Figure 4.6, two vectors, shown as arrows, point towards each other to model a parallel gripper partially closing. A grasp contact assumes its location is graspable for every point along the vector. Vector length generalizes the grasp type for different object sizes. Long vector lengths correlate with more object size variation while short vector lengths correlate with less object size variation (i.e. a grasp type becomes more size-specific). However, too long a vector length may generate larger pose offsets from an object's

primary axis; if closing a gripper uniformly, fingers will contact the object at different times causing a sliding motion. $\mathbf{V_g}(i,j,k)=0$ represents empty space. $\mathbf{V_g}(i,j,k)=-255$ for constraints. A grasp type's contact points and constraints are stored in a point cloud prior to voxel grid insertion. To indicate a point's positive or negative value, its RGB (green and red) colour value is changed.

Optionally, it is desirable to incorporate additional constraints with the grasp type model; their purpose is to create desirable behaviour for a grasp type. A collision-free linear path to grasp an object can be discovered by cross-correlating voxel grid grasp types with partial object voxel grid representations. Referring to Figure 4.6, constraints outside the gripper contacts penalize surfaces larger than the gripper's "maximum opening", specify a minimum gap required to place a finger between objects, and center the gripper palm with respect to contact points (that coupled actuation would need). A wrist constraint prevents the gripper's palm from colliding with an object's surface. An additional contact vector from the palm can be added to favour grasp poses along an object's observed surface. This contact requires the gripper's palm to face an object's observed surface. The voxel representations in Figure 4.6 and Figure 4.10 demonstrate this contact vector for lateral and tripodal grasp types. Without this vector, cross-correlation can yield valid pose results from both sides of an object (on observed and unobserved sides), and for safety, grasp poses should only exist in observed regions.

Frames {**o**} and {**g**} are attached to the center of $\mathbf{V_o}$ and $\mathbf{V_g}$ bounding boxes respectively. Object and grasp type voxel grids are built within their respective reference frames; this allows points assigned to $\mathbf{V_g}$ to be rotated first and registered to $\mathbf{V_o}$ afterwards. The object's point cloud updates $\mathbf{V_o}$ after each scan. Cross-correlation between $\mathbf{V_g}$ and $\mathbf{V_o}$ is performed using the Fast Fourier Transform (FFT). Once voxel grid size and resolution are set, correlation runtime is fixed. Large object point clouds do not negatively impact our algorithm because they are down-sampled to a fixed size voxel grid. In Cartesian space, cross-correlating identical *M*-sized voxel grids $\mathbf{V_g}$ with $\mathbf{V_o}$ at any gripper orientation $\mathbf{R}(\alpha_g,\beta_g,\gamma_g)$ is defined as:

$$\mathbf{V_{go}}_{(\alpha,\beta,\gamma)}(x,y,z)=\sum_{i,j,k=0}^{M}\mathbf{R}_{(\alpha,\beta,\gamma)}\mathbf{V_g}(i,j,k)\,\mathbf{V_o}(x+i,y+j,z+k)$$

Since a voxel grid represents physical space, a grasp type model $\mathbf{V_g}$ is bounded by a relatively small box while the object voxel is adjustable. The largest impact to this algorithm is voxel resolution (V_{res}), but for grasping, resolution can be about as coarse as a gripper's finger width. Correlation also solves two problems with one step: 1) it indicates where a grasp type shape is most similar to the object, and 2) the wrist constraint length determines a collision-free linear path for the gripper to move through. A complete gripper pose, using orientations from Stage 1, is found for an *N*-contact grasp type when any voxel $\mathbf{V_{go}}_{(\alpha,\beta,\gamma)}(x,y,z)$ is greater than or equal to *N*.

4.6.5. Stage 2b: Verifying Normals and Contacts

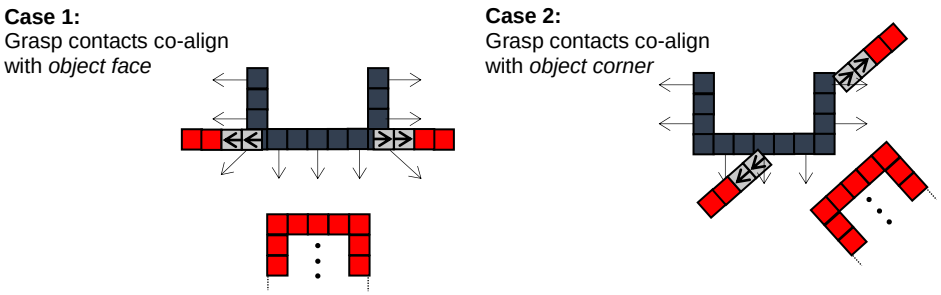


Figure 4.7: Removing Illogical Grasps when Surface and Contact Normals Mismatch. Contact location (light green) examples for a Partially Scanned Box (blue).

Since shape and scale matching are decoupled into sequential steps, the results after cross-correlation may include locations that do not logically yield a grasp. Two examples are shown in Figure 4.7; Case 1 shows a grasp type's contacts creating a plane in the figure. As a result, the correlation will correspond to a planar surface on one object face instead of two opposing faces. A single face on an object is not graspable. Even if the solution appears graspable, contacting edges are not desired because these locations offer the least amount of surface area for the gripper to touch. Case 2 in Figure 4.7 shows a similar example, but contacts align at an object's corner (i.e. several faces that are not opposing). If the gripper contacts close around an object's corner, the object will likely slip free.

Verification reasonably checks grasp contacts and their respective normal so that both match the partially scanned object surface. A grasp type position is verified by re-checking the contact normal's direction. At every potential grasp position, a k-nearest neighbour search is performed for each contact relative to the object(s)[87]. When the closest point on the object is discovered, the inner product of its normal with the gripper

normal is taken. The inner product for verification must be greater than a parameterized threshold (n_{th}). We define it as a similar size as one surface normal histogram bin, i.e. $n_{th} = \cos(\Delta\gamma)$.

4.6.6. Additional Stages: Task-based Grasping

A strength for the first two stages is many reasonable grasp poses are generated in near real-time by matching a grasp type to an observed object's surface. This is desired for a robotic platform experiencing uncertainty because many alternative grasp pose options exist if the desired final grasp pose fails. In addition, more options permit higher-level decisions to refine and select a grasp location based on a specified task.

For robot-to-human object transfer, a robot should select grasp pose locations that permits more surface area for another person to grasp; this action gives a person more opportunities to hold an object securely. However, estimating total surface area is difficult given a partial object model. Instead, overlapping grasp poses from all grasp types can identify graspable locations easily. For example as depicted in Figure 4.8, a cylinder's center has more grasp pose solutions than the cylinder's ends. If many grasp pose solutions exist at one region that match several grasp types, that region is assumed easily graspable. An algorithm can select this region if a task is specified as securing an object. For a robot-to-human transfer task, an algorithm can avoid this region, leaving it for the human receive to grasp.

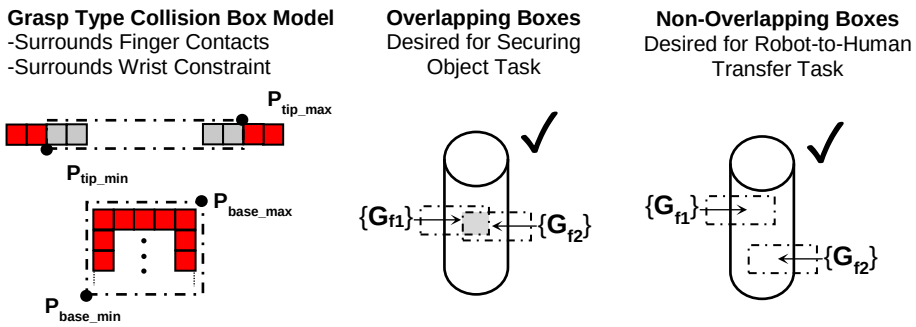


Figure 4.8: Task Selection based on Box Collisions

Given grasp type point clouds used to create grasp type voxel models depicted in Figure 4.6 and Figure 4.8, collision boxes encapsulate gripper contacts and wrist constraints within one voxel resolution. The gripper contact collision box detects if finger contacts from two grasp poses overlap. The wrist contact collision box detects if two grasp poses

are orientated in similar directions. The gripper contact collision box is defined using two points, denoted as $\mathbf{P}_{tip\_min}$ and $\mathbf{P}_{tip\_max}$, that surround the minimum and maximum gripper contact point (by one voxel resolution). Similarly, the wrist constraint collision box is defined using two points, denoted as $\mathbf{P}_{base\_min}$ and $\mathbf{P}_{base\_max}$, that surround the minimum and maximum wrist penalty locations. In practice, eight points define the edges of a collision box, and all eight points are transformed (i.e. rotated and translated) in the world frame. After transformation, the minimum and maximum edges are assigned $\mathbf{P}_{name\_min}$ and $\mathbf{P}_{name\_max}$ respectively. These points create two bounded regions for all other grasp type poses to avoid. Iteratively, collision boxes are created similarly for all remaining grasp poses to test overlap with these regions.

By labelling two collision boxes as ‘cb1’ and ‘cb2’ respectively, collision boxes overlap along one axis if $x_{cb1\_max} \geq x_{cb2\_min}$ and $x_{cb1\_min} \leq x_{cb2\_max}$. If this test is repeated for the remaining axes, every test must pass to determine if $cb1 \cap cb2$; if any test fails, cb1 and cb2 do not overlap. The following ranks are applied for robot-to-human transfer and securing an object tasks:

- **Robot-to-Human Transfer Task**: increment rank every instance two collision boxes do not overlap, shown in Figure 4.8.
- **Securing an Object Task**: increment rank every instance two collision boxes overlap, shown in Figure 4.8.

Intuitively, the robot-to-human transfer rank maximizes when both finger and wrist collision boxes avoid all other collision boxes, and securing an object task maximizes when both finger and wrist collision boxes overlap with all other collision boxes.

4.6.7. Calibrating (or Offsetting) a Gripper Frame for each Grasp Type

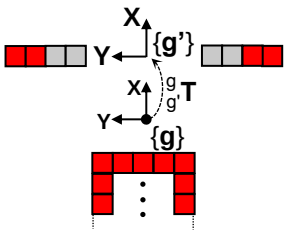


Figure 4.9: Shifting a Grasp Frame to tune Grasping

The grasp gripper frame (i.e. **g**) may not be ideally located at the desired grasp location between desired contact locations, as shown in Figure 4.9. A gripper model can be

recreated to locate grasp contacts symmetrically around a new frame offset, assuming the same grasp type; alternatively, the gripper frame can be transformed to a new location. Given the grasp type frame's origin (i.e. ${}^{W}_{g}\mathbf{P}_{org}$), an intermediate calibration transform (i.e. ${}^{g}_{g'}\mathbf{T}$) or offset (i.e. ${}^{g}_{g'}\mathbf{P}$) is defined relative to the gripper frame to create a new grasp type origin (i.e. ${}^{W}_{g'}\mathbf{P}_{org}$). The new grasp type origin and pose are defined as:

$$ {}^{W}_{g'}\mathbf{P}_{org} = {}^{W}_{g}\mathbf{T}\,{}^{g}_{g'}\mathbf{P} + {}^{W}_{g}\mathbf{P}_{org} $$

$$ {}^{W}_{g'}\mathbf{T} = {}^{W}_{g}\mathbf{T}\,{}^{g}_{g'}\mathbf{T} $$

Heuristically Removing Grasp Poses Passing Through a Table Surface

Within the grasp pose generation algorithm, the options available to avoid and mitigate generating a grasp pose that causes a gripper to collide with a table's surface is as follows: 1) Grasp type shapes should not be generalized to cross-correlate with a table's surface, 2) a Stage 1 orientation search can define pitch (i.e. β) to only orientate towards (i.e. β:[-π/2,0]) and not away (i.e. β:(0,π/2]) from a table's surface. For example, grasp poses for a box can orientate underneath it, and 3) grasp type model voxel constraints, shown in Figure 4.6, can define finger limits or added constraints around desired contact locations to detect when a grasp type gripper model passes through a table's surface. Within our proposed grasping system, octree $\mathcal{W}$ guarantees a mobile manipulator avoids collisions with any observed obstacle and unobserved regions. However, in the absence of these measures, grasp pose generation after cross-correlation (i.e. Stage 2) can be heuristically filtered.

Previously discussed in Section 4.6.7, the grasp type gripper frame can be offset to a new location relative to its original frame. Complete pose information is given by referencing locations relative to the final grasp frame (i.e. relating to Section 4.6.7, ${}^{W}_{g}\mathbf{T}={}^{W}_{G_f}\mathbf{T}$ is the final grasp pose and ${}^{g}_{g'}\mathbf{T}={}^{G_f}_{G_f'}\mathbf{T}$ represents any pose relative to the final grasp frame). For every grasp pose generated after cross-correlation, edges relative to the grasp frame (i.e. ${}^{G_f}_{G_f'}\mathbf{T}$) are defined to create a collision boundary for the gripper. Iteratively, eight edges locate four corners bounding the gripper's wrist and four corners bounding the gripper's fingertips. If any corner is located below a table surface plane, the grasp pose is ignored. Table surface height can be observed from a point cloud using planar model segmentation[88, 89] or assumed known a priori. Determining if a

boundary is below a threshold is less computationally expensive than sending grasp poses to a trajectory planner, calculating inverse kinematics, and determining if the mobile manipulator collides with observed obstacles. For uneven or unknown surfaces, utilizing a collision map, e.g. octree $\mathcal{W}$, is more robust for any planning algorithm because a table is represented as an avoidable region without using prior assumptions.

4.7. Experimental Setup

4.7.1. Implementation

Generalizing multi-grasp type pose generation is developed as a sub-system within the Robot Operating System (ROS) in our lab for a “fetch an object” task. Our mobile manipulator comprises of a 3-DOF base (Powerbot), a 6-DOF manipulator (Schunk Power Cube arm), and a 7-DOF 3-fingered Schunk Dextrous Hand (SDH). A Hokuyo URG-04LX planar laser is mounted on the manipulator’s wrist as an eye-in-hand sensor and scans all objects in the environment[52]. Its angular resolution is 0.36°. The overall integrated system comprises a motion planning and next best view (NBV) algorithm for automatically scanning an unknown object of interest in an unknown environment and is discussed in detail[52]. The focus of this paper is to show reasonable and successful grasp poses for different grasp types can be generated for a partially or fully scanned objects. The FFT algorithm that cross-correlates voxels is developed using the FFTW library[90]. Cross-correlation is performed by evaluating the Fourier transform for the object and conjugated (i.e. reversed) gripper voxel grids and evaluating the inverse Fourier transform after their product (i.e. $\mathbf{V_o} \star \mathbf{V_g} = \mathcal{F}^{-1}\left\{\mathcal{F}[\mathbf{V_o}(t)]\mathcal{F}\left[\mathbf{V_g^*}(-t)\right]\right\}$). Experiments are conducted using an Intel Core i5-3210M CPU @ 2.5 GHz and 16 GB of RAM.

Hardware Limitations

Grasp execution engages all motors simultaneously to trap an object. Sliding may occur for our system implementation because tactile feedback is not implemented to stop motors once they contact the object. To mitigate bumping or sliding, other grasping works added a subsequent stage to rank grasp poses based on grasp robustness[81], re-align the gripper’s palm so all fingers contact an object simultaneously[10, 41], or stabilize a grasp from an uncertain pose using tactile feedback[91].

4.7.2. Grasp Models and Parameters

For all experiments, three grasp types (i.e. lateral, tripodal, and power), shown in Figure 4.10 are repeatedly searched at different voxel resolutions. A lateral grasp is modelled like a parallel jaw gripper; distal pads (or fingertips) move towards each other in a pinching motion. A tripodal grasp is similar, but each fingertip is separated by 2π/3,

forming a triangular shape that closes. A power grasp is modelled as a diamond that encloses proximal and distal pads around an object at ±π/6. When modelling precise proximal/distal motion for a power grasp, grasp poses only appeared for a small range of object sizes. This merits further investigation.

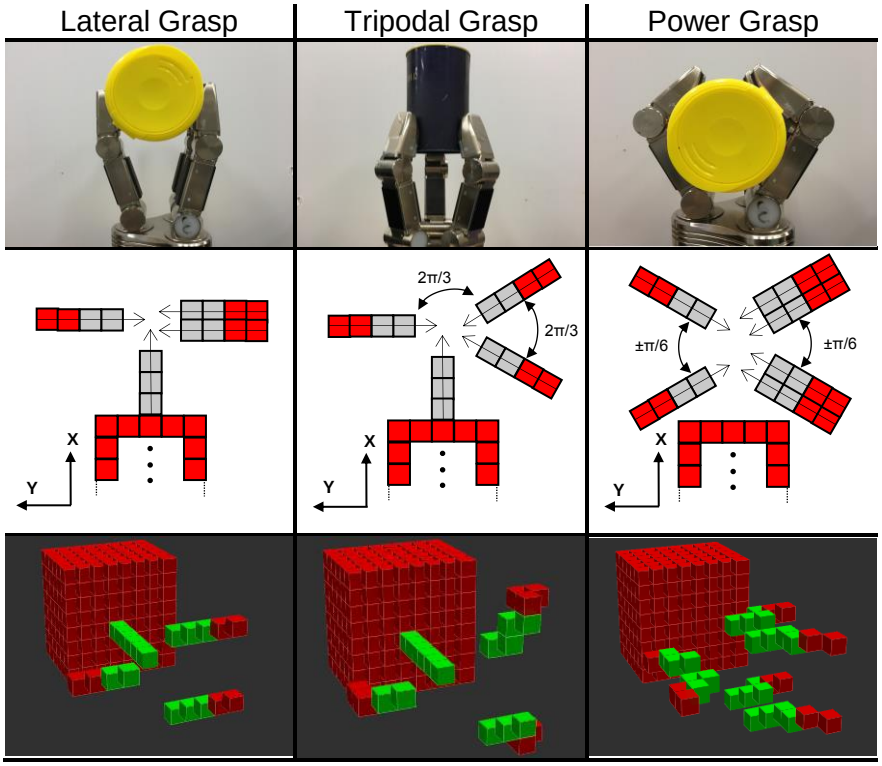


Figure 4.10: Gripper images for three grasp types (top row), voxel models for the grasp types (middle row), and implemented voxel representation (bottom row)

The total contact vector length to model our gripper's motion in voxel grid $\mathbf{V_g}$ is 5.25cm, where 3.75cm is finger motion length, i.e. d_f, applied as a positive value (i.e. $\mathbf{V_g}(i,j,k)=0$) and 1.5cm is finger spacing length, i.e. d_s, dedicated as a constraint (i.e. $\mathbf{V_g}(i,j,k)=-255$). These vectors are separated to generalize each grasp type to discover correlations for objects that range between 3.5cm to 11.0cm in diameter. A fourth contact vector from the palm is added to the lateral and tripodal grasp to favour grasp poses along the scanned object's observed side. The power grasp does not have this vector because most of the object needs to be observed before this grasp type is discovered. The wrist constraint is 12.0cm wide (i.e. Schunk SDH width) and extends 10.0cm in length. This length guarantees the gripper moves collision-free 10.0cm along a linear trajectory prior to reaching the final grasp pose. Proximal joint motors engage at a constant velocity to

apply a lateral and tripodal grasp; both proximal and distal joint motors engage to complete a power grasp.

Table 4.1 summarizes all voxel model and grasp generation algorithm parameters selected for experimentation. Behaviour from our grasp pose generation algorithm is affected by four parameters: 1) Normal histogram search Euler angles, i.e. α, β, γ, 2) normal histogram resolution, i.e. $\Delta\alpha\beta\gamma$, 3) voxel resolution, i.e. $\mathbf{V_{res}}$, and 4) verification angle threshold, i.e. n_{th}. Normal histogram search angles and resolution affect the quantity of gripper poses to query during Stage 1. Brute force or heuristics can select search angles. For example, search angles for integrated grasping experiments are heuristically selected to be parallel to the ground plane because the mobile base will not approach an object close enough to allow any alternative approach. Normal histogram resolution is below 20° to mitigate grasp failures due to pose orientation error[70, 71]. In Stage 2, voxel resolution, $\mathbf{V_{res}}$, affects the volume a voxel 'smooths' and object's point cloud and respective surface normals. The final verification angle threshold, n_{th}, rejects grasp poses when the length between any finger contact normal and object surface exceeds this value.

The object and gripper voxel size are defined as $\mathbf{V_o}$= 50x60x30cm and $\mathbf{V_g}$=30x30x30cm respectively. The object voxel encapsulates all objects in the world frame. For each grasp type, the gripper wrist rolls α:[0,2π), pitches (i.e. pivots up/down) β:[-π/2,π/6], and yaws (i.e. rotates around the object) γ:[0, 2π) at $\Delta\alpha\beta\gamma=\pi/6$ increments, unless stated otherwise. After cross-correlation, a verification threshold $n_{th}=0.9*\cos(\pi/6)$ is chosen to confirm the object surface normal align with gripper contact normal within the resolution of the surface normal histograms. A maximum of 576 cross-correlations could be performed, but in practice, fewer cross-correlations (i.e. ~20%) are performed because normal histograms $\mathbf{H_o}$ and $\mathbf{H_g}$ remove orientations where a gripper normal does not exist on the object's surface.

Table 4.1: Summary of Gripper Model and Grasping Algorithm Parameters for Experiments

Voxel Modelling	**Symbol**	**Value**
Object Voxel Size	$\mathbf{V_o}$	50x60x30cm
Gripper Voxel Size	$\mathbf{V_g}$	30x30x30cm
Finger Motion Length	d_f	3.75 cm
Finger Spacing Length	d_s	1.50 cm
Palm Width	N/A	12.0 cm
Straight Line Trajectory Motion	N/A	10.0 cm
Grasp Algorithm Experiments	**Symbol**	**Value**
Normal Histogram Search	α, β, γ	α:[0,2π), β:[-π/2,π/6], γ:[0, 2π)
Normal Histogram Resolution	$\Delta\alpha\beta\gamma$	π/6 or π/12 radians
Voxel Resolution	V_{res}	Variable, Typically: 1.0 – 1.5 cm^3
Verification Angle Threshold	n_{th}	0.9*cos(π/6)
Integrated Grasping Experiments	**Symbol**	**Value**
Normal Histogram Search	α, β, γ	α:[0,2π), β:{-π/2 ,[0, π/6]}, γ:[0]
Normal Histogram Resolution	$\Delta\alpha\beta\gamma$	π/6 rad
Voxel Resolution	V_{res}	1.5 cm^3
Verification Angle Threshold	n_{th}	0.9*cos(π/6)

4.8. Experimental Results

Experimental results are divided into two categories: 1) grasping algorithm behaviour as parameters change and 2) fully integrated grasping. The first category utilizes simulation, simulated data, and household objects from the YCB dataset[92] to visualize our grasping algorithm's performance as key parameters change. The second category characterizes the integrated system by grasping and lifting real objects from partially scanned models while experiencing uncertainty. Grasp poses are displayed as three different colour arrow markers for each grasp type, magenta for lateral, yellow for tripodal, and red for power.

4.8.1. Grasping Algorithm Performance as Parameters Change

Experiment 1: Computation Time as Voxel Grid Size, V_{res}, Increases

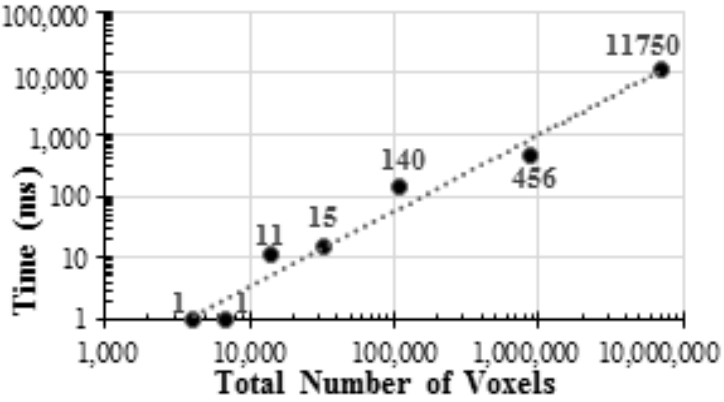


Figure 4.11: FFT-based Correlation Computation Time
as a function of total number of Voxels.

A tin can (Dimension: 10.5cm x 22.0cm) and hand drill (19.0cm x 6.5cm x 22.0cm) are completely scanned and modelled a priori with 30,438 points with a resolution of 3.0mm. The point cloud model is loaded into our algorithm and executed for different voxel resolutions, where $\mathbf{V_{res}}$={0.25, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0}cm^3, and FFT computation time is averaged. The major computation time is FFT-based correlation in Stage 2 and is shown as a function of total voxels in Figure 4.11. Time is linear with respect to total number of voxels (with a slope of 1) but exponential to dimension D, i.e. D = 3. Our system can process near real-time grasp results for voxel grid sizes up to 800,000 voxels on a standard laptop CPU.

Experiment 2: Pose Results for Simulated Objects and YCB Model Dataset

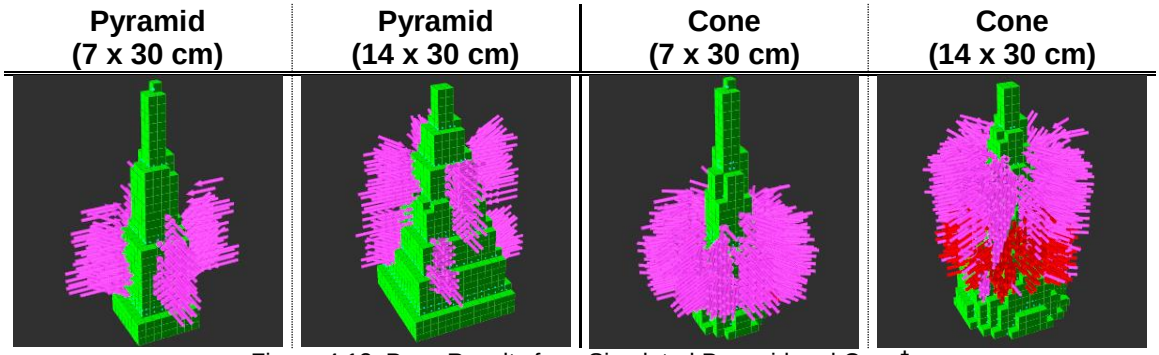


Figure 4.12: Pose Results for a Simulated Pyramid and Cone†
† Lateral (magenta) and Power Grasp (red)

A simulated pyramid and cone point cloud is inserted into our algorithm's object voxel grid. As shown in Figure 4.12, the grasp poses found by our algorithm are poses broadside to the objects, and as their base size increases, poses migrate upwards to match their desired size. Tripodal grasps are not found due to the wrist constraint preventing a top-down grasp, and power grasps only appear when its desired radius on the object exists.

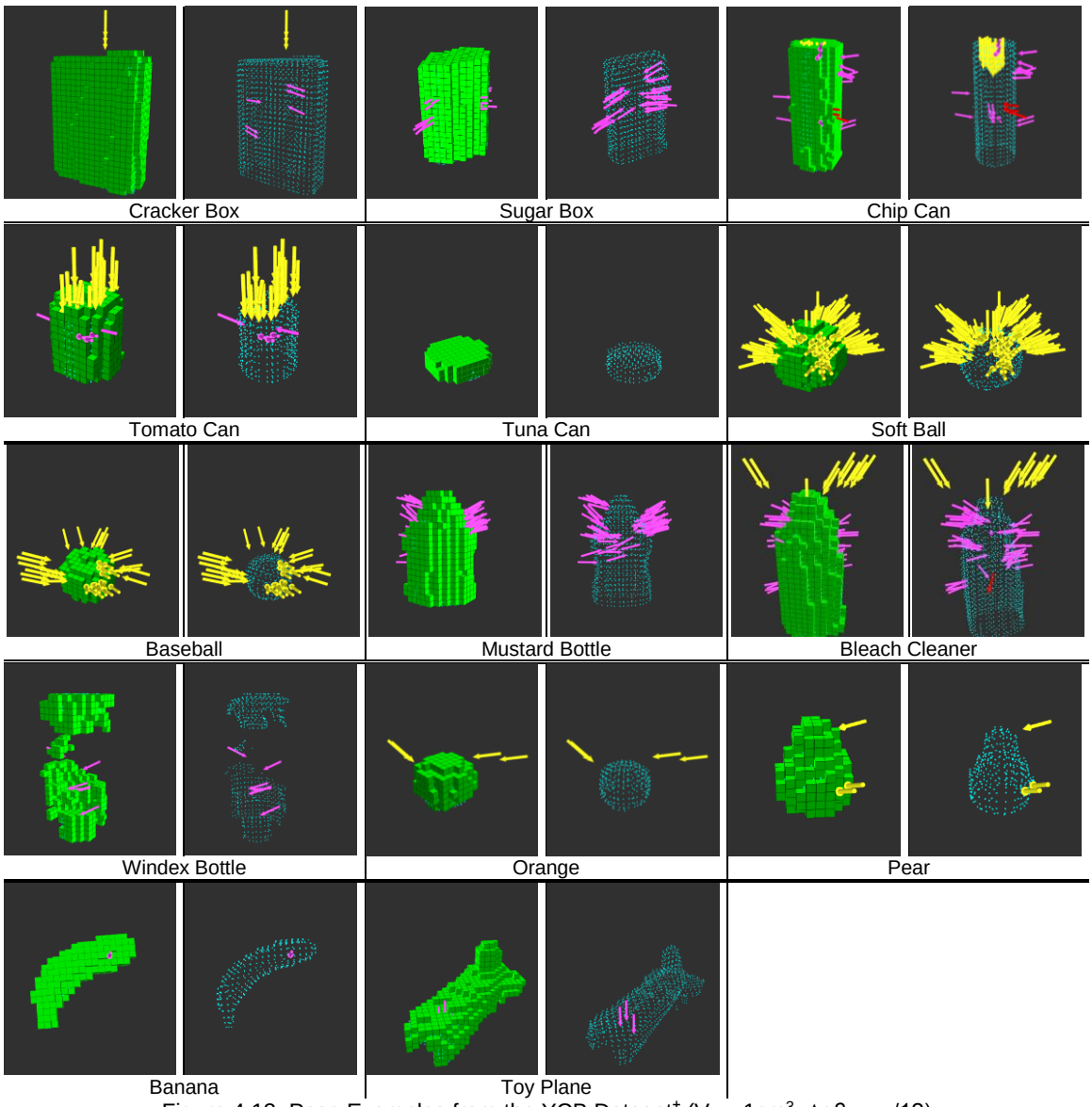


Figure 4.13: Pose Examples from the YCB Dataset† (V_{res}: 1cm^3, $\Delta\alpha\beta\gamma$: $\pi/12$)
† Lateral (magenta), Tripodal (yellow), and Power Grasp (red)

Figure 4.13 visualizes and Table 4.2 summarizes pose results for household objects from the YCB model dataset after varying voxel resolution (Vres) and normal histogram bin size ($\Delta\alpha\beta\gamma$). We make the following salient observations:

- Grasp type shape associates consistently with similar object shapes.
- Fine voxel resolution is more appropriate for smaller objects.

Lateral grasps are associated to box-like surfaces, tripodal to cylinders/balls, and the power grasps to cylinders. For more complex objects, like the mustard and Windex bottles, lateral grasps are found along the base's principal axis and top-down grasps are proposed to grab the mustard's base or Windex bottle's head. The algorithm worked

nicely even if the bottle's model is partial, not revealing its entire shape. Experiments also reveal fine voxel resolution is needed for smaller objects, like the banana or tuna fish can. Large voxels applied to small objects average too many surface normals, making normal estimates inconsistent relative to their surface.

Table 4.2: Pose Generation Results for YCB Model Dataset[¥]
[¥] Bold indicates most frequent grasp type discovered

Grasp Type:	**L/ T/ P**	**L/ T/ P**	**L/ T/ P**
Settings	V_{res}: 1cm^3 $\Delta\alpha\beta\gamma$: π/6	V_{res}: 1cm^3 $\Delta\alpha\beta\gamma$: π/12	V_{res}: 0.75cm^3 $\Delta\alpha\beta\gamma$: π/6
Cracker Box	**5**/ 3/ 0	**9**/ 3/ 0	**67**/ 43/ 0
Sugar Box	**7**/ 0/ 0	**25**/ 0/ 0	**66**/ 7/ 0
Chips Can	15/ **27**/ 4	**40**/ **43**/ 7	**152**/ 36/111
Tomato Soup	2/**40**/ 0	11/**40**/ 0	13/ **78**/ 0
Tuna Fish Can	0/ 0/ 0	0/ 0/ 0	0/ **13**/ 0
Softball	0/ **133**/ 0	0/**139**/ 0	2/**292**/ 0
Baseball	0/ **42**/ 0	0/ **43**/ 0	1/**202**/ 0
Mustard Bottle	**22**/ 1/ 0	**64**/ 0/ 0	104/**150**/ 8
Bleach Cleanser	12/ **41**/ 2	**58**/ 15/ 3	**152**/ 88/ 57
Windex Bottle	1/ **15**/ 0	**7**/ 0/ 0	**6**/ **4**/ 0
Orange	0/ **1**/ 0	0/ **4**/ 0	0/**392**/ 0
Pear	0/ **11**/ 0	0/ **3**/ 0	16/ **37**/ 0
Banana	**1**/ 0/ 0	**1**/ 0/ 0	**14**/ **11**/ 0

Experiment 3: Pose Results as Voxel Resolution Changes

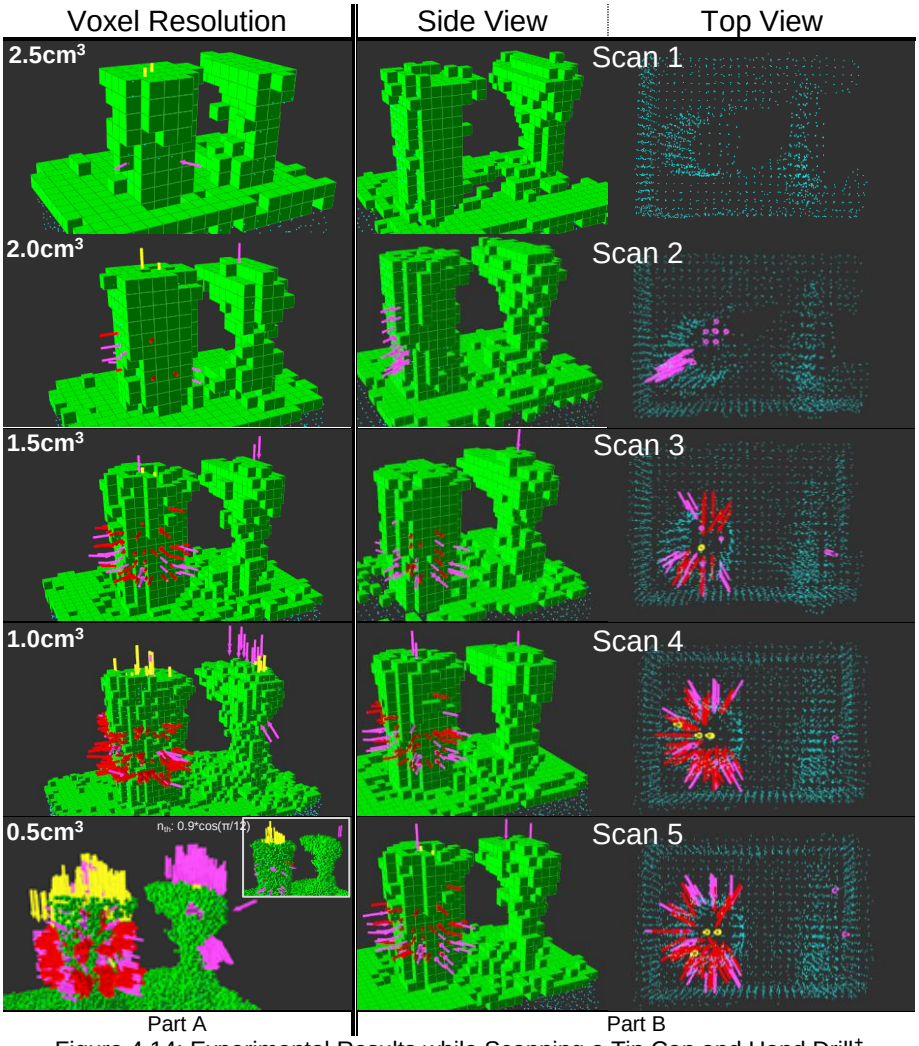


Figure 4.14: Experimental Results while Scanning a Tin Can and Hand Drill†

† Lateral (magenta), Tripodal (yellow), and Power Grasp (red)

Figure 4.14a visualizes grasp results for voxel resolutions $\mathbf{V_{res}} = \{0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}\text{cm}^3$ from the previous experiment. Magenta, yellow, and red arrows indicate a grasp pose (normal to the gripper palm) for lateral, tripodal, and power grasps respectively. We make the following salient observations:

- Finer resolution reveals more details and more grasp poses.
- Reasonable solutions are consistently found for low and high voxel resolutions.
- The wrist constraint prevented poses to generate where a straight line grasp trajectory would pass through a neighbouring object.

In general, higher resolution reveals more details from the scanned objects and more grasp poses are discovered. Interestingly, reasonable grasp solutions for all grasp types are consistently found at both low ($\mathbf{V_{res}} = 2.0\ \text{cm}^3$) and high ($\mathbf{V_{res}} = 0.5\ \text{cm}^3$) resolutions.

This was also observed for the YCB model dataset. For example, a tripodal grasp is available above the tin can to grasp downwards, the drill can be grasped from above, and all lateral/power grasps along the tin can's side face away from the drill to avoid collision. In fact, the grasping system autonomously selected these pose and avoided collision with the cordless drill; autonomous grasp examples are shown in Figure 4.15. Figure 4.15 also visualizes the real scene scanned to create models presented in Figure 4.14.

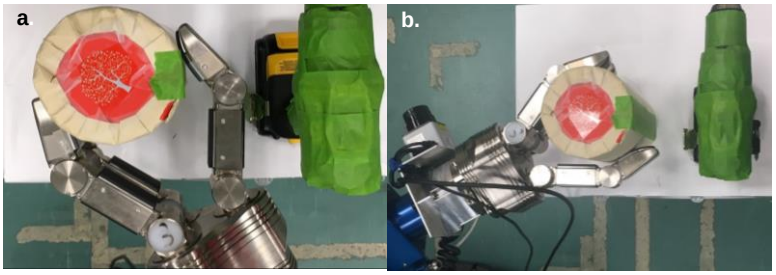


Figure 4.15: Autonomous Grasp Examples that avoids Collisions with a Neighbouring Object

When $\mathbf{V_{res}} = 0.5\ \text{cm}^3$, noise (and more details) causes some of these top-down tripodal grasps to be offset from center. To mitigate these offsets, the contact vector length can be reduced or verification threshold can become more specific. A verification threshold example is shown in Figure 4.14a by decreasing the normal verification angle from π/6 to π/12 for $\mathbf{V_{res}} = 0.5\ \text{cm}^3$. Resolutions $\mathbf{V_{res}} = \{1.0, 1.5\}\text{cm}^3$ smooth noise from the point cloud and clearly select top-down tripodal grasps. On the other hand, low resolution may introduce a physical position 'offset' error. Our grasp strategy is to trap the object between gripper fingers. A small position error (i.e. < 1.0cm) is likely to be relatively harmless for grasping. However, larger errors may cause one finger to bump into the object first, possibly resulting in the object being moved, e.g. sliding or even tipping over, potentially causing a failed grasp; this problem is avoided by utilizing tactile sensing to stop finger motion once a tactile pad contacts an object.

Experiment 4: Pose Results from Incomplete Information

Figure 4.14b shows grasp results while scanning an object from five different viewpoints using $\mathbf{V_{res}} = 1.5\ \text{cm}^3$. Scans are taken counter-clockwise, approximately π/4 radians apart, around the objects shown in Figure 4.14b. Each consecutive scan is registered and merged with the previous until a complete object point cloud is created; more details about this process can be found from our previous work[52]. We make the following salient observation:

- Grasp pose generation consistently selects similar poses, regardless of a object model's completeness.

The first scan did not generate grasp poses. This is expected because a parallel grasp needs two opposing surfaces to be observed to generate a result. The second scan in Figure 4.14b demonstrates this behaviour as lateral grasps are found top-down and along the tin can's side. All grasp types can be found by the third scan; at this point, the objects' three sides are observed. In Figure 4.14b, the point cloud is experiencing 3cm of registration error; this can be observed from the fifth scan, looking at the point cloud's top-right hand corner, where the corners do not align. Please note that our point cloud is down-sampled to the same resolution as $\mathbf{V_{res}}$. This error does not significantly affect our algorithm. Pose locations are still centered with respect to each object, and can allow the gripper to trap the object between its fingers.

Experiment 5: Visualizing Task-Based Grasp Poses

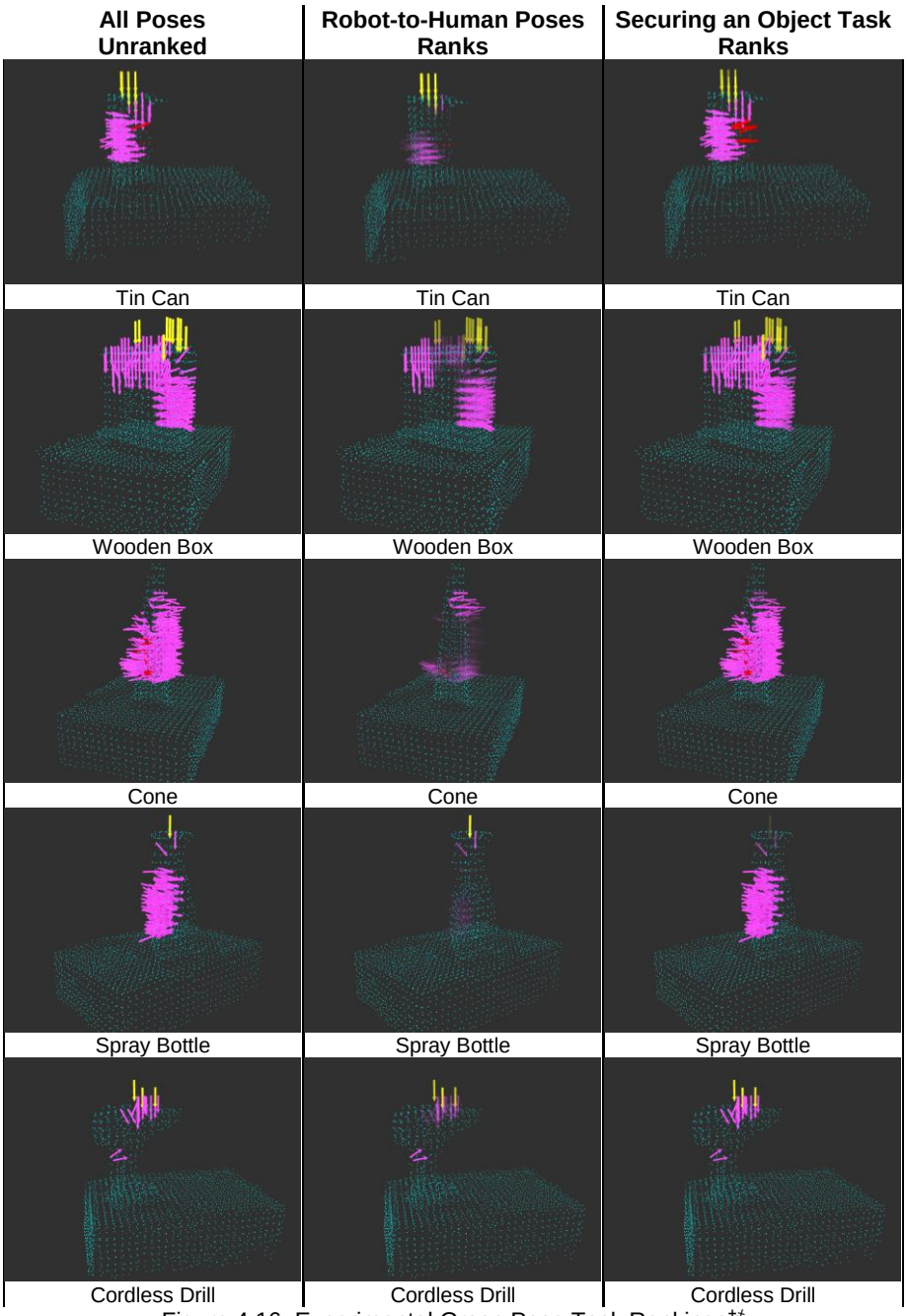


Figure 4.16: Experimental Grasp Pose Task Rankings†≠

† Lateral (magenta), Tripodal (yellow), and Power Grasp (red)

≠ High Ranks (Opaque), Low Ranks (Transparent)

The purpose for this experiment is to demonstrate how higher-level decisions can be incorporated with grasp pose generation to rank results. Grasp pose selection is presenting considering two different tasks: 1) grasping an object to transfer to a human receiver, and 2) grasping an object to hold it. Object point clouds from Figure 4.13 and

Figure 4.18 are re-used, and grasp poses for all grasp types are searched for α:[0,2π), β:[-π/2,π/6], and γ:[0] at ∆αβγ=π/12 increments. Magenta, yellow, and red arrows indicate a grasp pose (normal to the gripper palm) for lateral, tripodal, and power grasps respectively. Unranked grasp poses are compared to poses ranked for robot-to-human transfer and securing an object. To visualize rankings, high ranking grasp poses appear more opaque while low ranking grasp poses (i.e. not a good location for robot-to-human transfer) appear more transparent to invisible.

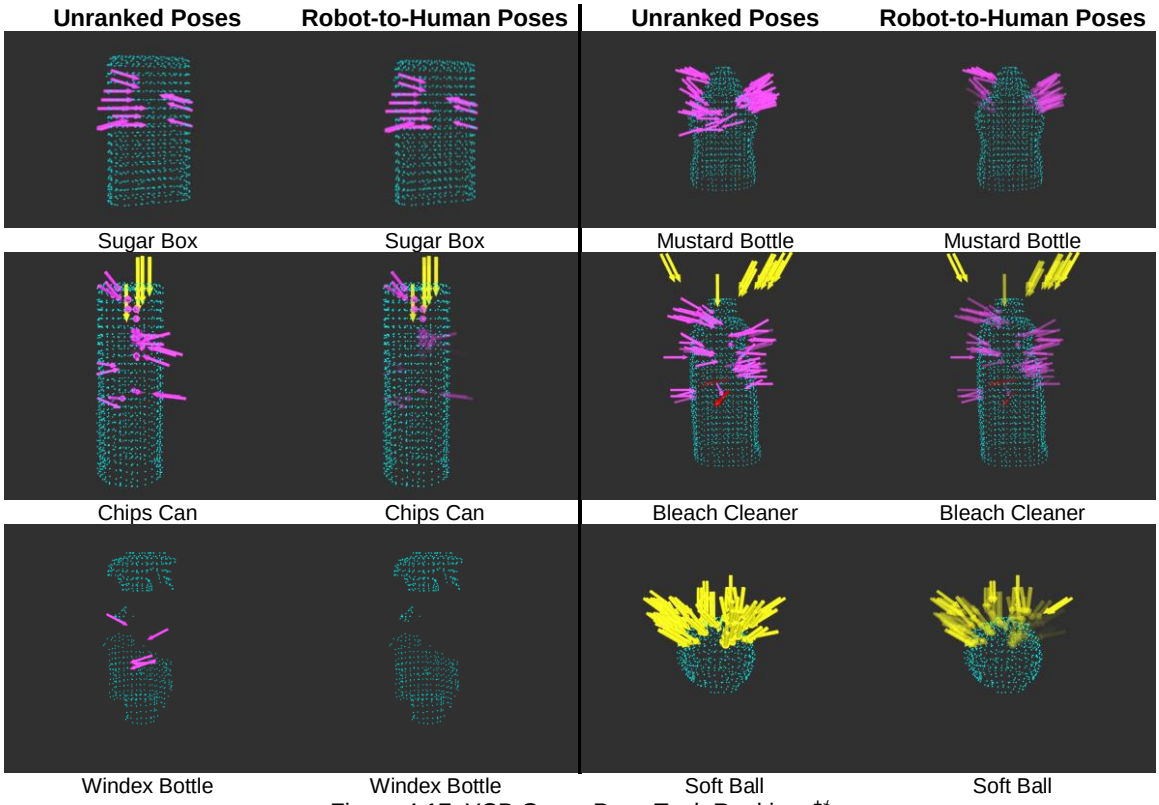


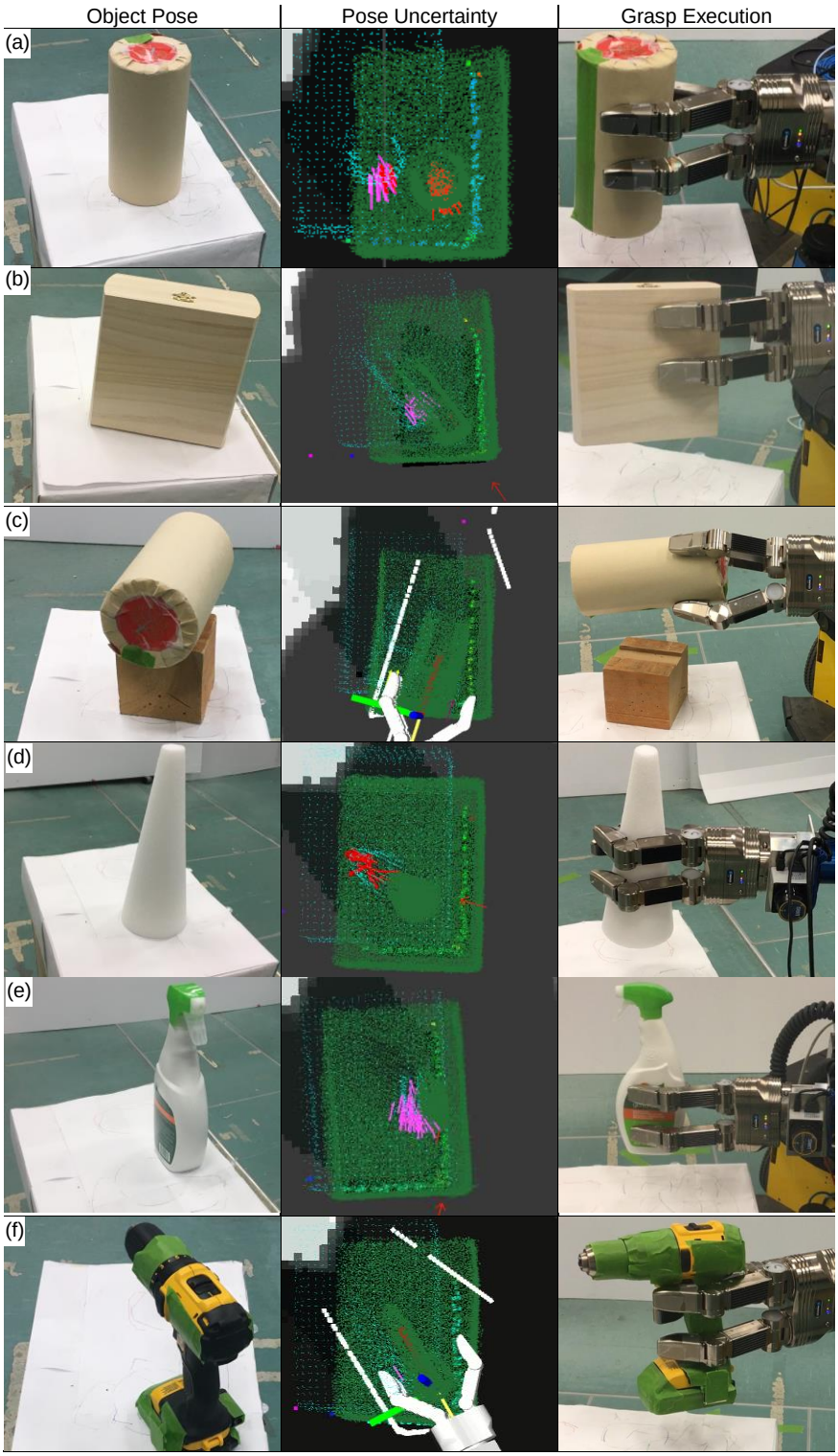
Figure 4.17: YCB Grasp Pose Task Rankings†≠
† Lateral (magenta), Tripodal (yellow), and Power Grasp (red)
≠ High Ranks (Opaque), Low Ranks (Transparent)

Comparing unranked poses to proposed task poses, Figure 4.16 and Figure 4.17, robot-to-human transfer poses prioritize that have less grasp overlap with other grasp poses. Securing an object prioritizes object locations having the most grasp overlap. If all grasp poses overlap (e.g. the Windex bottle), no pose for robot-to-human transfer is discovered. The soft ball shown in Figure 4.17 favours grasp poses further away from the object for transfer. This result is promising because more surface area around the ball would be available if finger tips, rather than fingers, grasp. However, the consequence is higher precision is necessary to grasp the soft ball without bumping it. Little change is observed from the sugar box example because grasp poses on either

side of the object do not interfere with each other; for transfer, either side can be grasped. A similar result is observed from the mustard bottle. For both the tin can and chips can, the can's top and base are ranked highest; the robot can grasp either location to allow a person to grasp the can's top or base for transfer.

4.8.2. Fully Integrated Grasping System Performance

Experiment 6: Grasp Execution for Real Objects



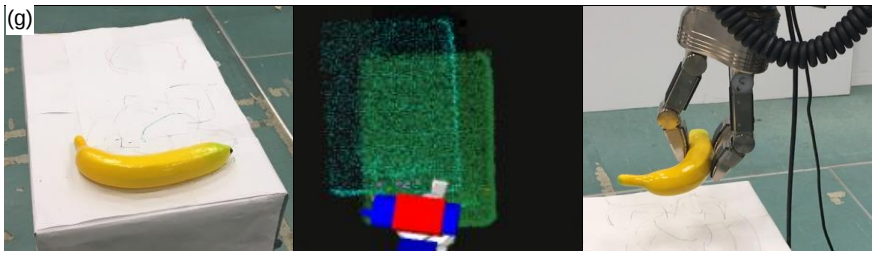


Figure 4.18: Executing Different Grasp Types with
Pose Uncertainty: Blue (Assumed), Green (Corrected).

Seven real-world objects are partially scanned, grasped, and lifted 10cm executing lateral, tripodal, and power grasp types discovered by our algorithm. Four objects represent primitive shapes while the remaining objects represent a tool, kitchen, and food items similar (in size, shape, and mass) to the YCB dataset [92, 93]. Objects sit at rest and poses are varied for each object, as shown in Figure 4.18. Our Hokuyo eye-in-hand sensor is not able to scan specular or black surfaces. Since several object surfaces are specular, objects are taped to create an observable Lambertian surface.

Each object was manually scanned counter-clockwise, approximately π/4 radians apart at four locations. Grasp poses for lateral, tripodal, and power grasp types using parameters summarized in bottom of Table 4.1. Due to a banana's small size and high curvature, no solutions were found at $\mathbf{V_{res}}$=1.5 cm^3; when resolution is increased to $\mathbf{V_{res}}$=0.75 cm^3, top-down lateral grasp types are found but correctly rejected by our algorithm due to fingers bumping into the supporting table. For this reason, the grasp frame for the lateral grasp type was translated 4 cm forward to the gripper's fingertips.

Our mobile manipulator system randomly selects among the grasp poses generated, checks if it corresponds to a reachable base pose (via inverse kinematics), and moves to a computed base position that partially corrects base pose uncertainty[52]. Grasp execution was repeated 10x per object. Other base locations were selected, and the experiment was repeated 2x; similar behaviour as reported was observed. A lift is considered successful if the object is raised 10 cm after grasping it, and a push indicates the object had a small sliding motion due to a finger pushing the object prior to grasping.

Table 4.3: Experimental Trials to Grasp Objects with Different Grasp Types[¥]
[¥] Bold indicates most frequent grasp type discovered

	Grasp Type	(Success)/ Trials	Lift	Push / Bump	Feasible **L / P/ T**	L^2 Error
Settings	**V_{res}: 1.5cm^3**		**α:[0,2π), β:[0], γ:[0]**			**∆αβγ : π/6**
Vertical Tin	Lateral	(10)/10	✓	✓	**56**/6/0	12cm
Vertical Tin	Power	(10)/10	✓	✓	56/**6**/0	12cm
Horizontal Tin	Tripodal	(10)/10	✓	✓	6/0/**3**	9cm
Box	Lateral	(10)/10	✓	✓	**84**/0/0	9cm
Cone	Lateral	(10)/10	✓	✓	**280**/9/0	9cm
Cone	Power	(10)/10	✓	✓	280/**9**/0	9cm
Spray Bottle	Lateral	(10)/10	✓	✓	**86**/0/0	5cm
Drill	Lateral	(10)/10	✓	✓	**1**/0/0	9cm
Banana	N/A	(0)/0	N/A	N/A	0/0/0	N/A
Settings	**V_{res}: 1.5cm^3**		**α:[0,2π), β:[-π/2], γ:[0]**			**∆αβγ : π/6**
Banana	Lateral	(10)/10	✓	✓	**5**/0/0	10cm

Table 4.3 summarizes results and shows our algorithm generates reasonable grasp pose solutions for partially observed objects. Our grasp type definitions successfully lifted objects without optimizing for dynamic stability. Given a banana's high curvature and small size, no grasps were found at $\mathbf{V_{res}}$=1.5 cm^3 because voxels average much of a banana's surface to compute accurate surface normals.

Finally, a note about uncertainties in our system. These arise due to: 1) mobile base odometry and kinematic arm model calibration affecting the gripper pose, 2) contact vector length causing positive correlation over a range, and 3) voxel resolution rounding a grasp position to a nearest voxel center. Uncertainty is corrected by a predefined final scan and is described in Section 3.6. This uncertainty is represented as L^2 Error in Table 4.3 and is visualized as the difference between green (i.e. corrected) and blue (i.e. original) point clouds under the "Pose Uncertainty" column in Figure 4.18). In addition, high curvature object shapes, e.g. a banana, are highly sensitive to contact locations.

Experiment 7: Complete Autonomous Modelling and Grasping Experiments

Three experiments are performed to demonstrate autonomous modelling, grasping, and lifting an unknown object in an unknown environment. A tin can, wooden box, and cone, shown in Figures 4.18a, 4.18b, and 4.18d are placed on a table to be lifted. Octree $\mathcal{W}$ and the mobile manipulator use the same resolution and tolerances as described in Section 3.7.1. Object modelling is guided using a 60cm x 60cm x 50cm octree $\mathcal{M}$ surrounding the OI with its resolution set to 0.8cm. Scan overlap is configured for ω = 0.4. Grasp pose generation is configured using parameters summarized in the bottom of Table 4.1. Fifty reservoir samples within a 30cm x 30cm x 10cm bounding box are tested to compensate for uncertainty reaching 15cm.

Three more experiments, using identical parameters as previously described, are repeated to model an unknown object and execute a power grasp. These tests are performed to intentionally extend the object modelling phase for the system and demonstrate autonomous execution of another grasp type. Generally, a lateral grasp is the first grasp type to be discovered because it requires fewer contact points (i.e. two opposing surfaces). To discover a power grasp type, more object surface area is observed because the grasp type wraps around an object.

During each experiment, the laptop's system clock records, Total Run Time, Base Planning Time, NBV Planning Time, and Pick Planning Time. Total Motion Execution Time is computed by taking the difference between Total Run Time and the sum of all other recorded times. Base Planning is time needed to create a base pose (for NBVs or objecting grasping). NBV planning is time needed to generate NBVs, update ranks, and predict IK at a projected base pose. Pick planning is time needed to predict a reachable base pose to grasp an object, all reservoir sampling, and Cartesian planning to the final grasp pose. Motion execution time is time taken to physically execute all base, manipulator, and gripper execution commands.

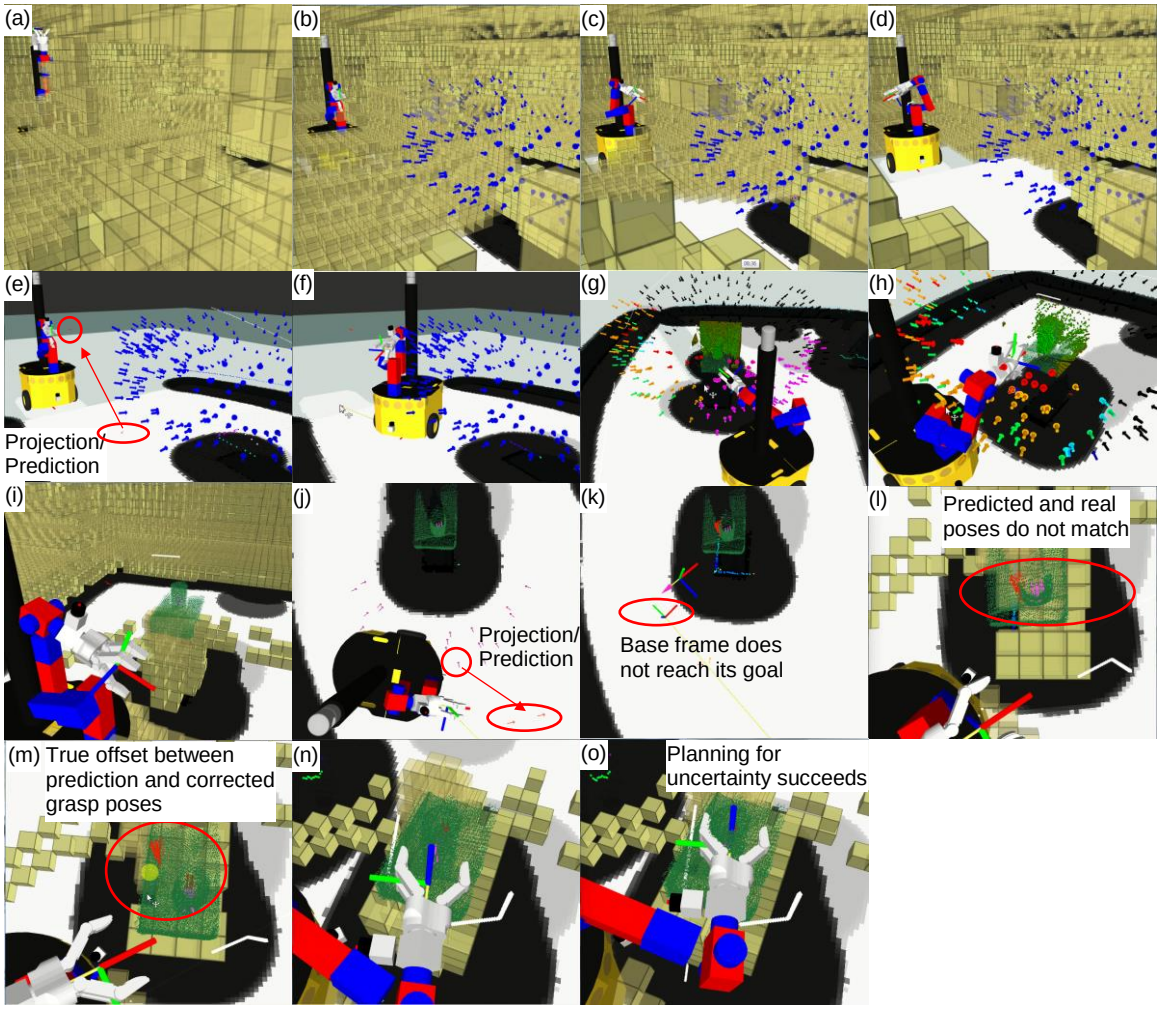


Figure 4.19: Experimental Results Autonomously Grasping an Unknown Tin Can Object†
† Lateral Grasp (magenta), Warmer coloured arrows have the highest ranks,
Yellow voxels are unknown (or unobserved) regions to avoid, Coloured Gripper Axis is Grasp Frame

Table 4.4: Time Taken to Autonomously Model and Grasp an Unknown Object using any Grasp Type

Object	# of NBV Scans	Final Pose Reselected?	Total Run Time	Total Motion Execution Time	Total Base Planning Time	Total NBV Planning Time	Total Pick Planning Time	L^2 error Compensated
	(#)		(s)	(s)	(s)	(s)	(s)	(cm)
Tin Can	2	No	810	613	0.002	11.496	169.324	15.4
Box	3	No	950	733	0.002	14.360	192.532	9.6
Cone	3	No	993	813	0.001	36.842	135.298	7.6

Table 4.5: Time Taken to Autonomously Model and Grasp of an Unknown Object using a Power Grasp

Object	# of NBV Scans	Final Pose Reselected?	Total Run Time	Total Motion Execution Time	Total Base Planning Time	Total NBV Planning Time	Total Pick Planning Time	L^2 error Compensated
	(#)		(s)	(s)	(s)	(s)	(s)	(cm)
Cone	3¥	Yes	915	807	0.002	53.449	43.408	10.6
Cone	3	No	1124	911	0.001	35.036	170.96	6.3
Cone	4	Yes	1181	1040	0.001	57.517	71.790	11.1

¥ One scan location failed. The mobile system moved to a new location to perform the final scan.

Table 4.4 and Table 4.5 summarize time needed to model and lift the unknown objects. Scanning an object, discovering a grasp pose, and executing a successful grasp that lifts an object 10 cm is performed within nineteen minutes. Primarily, the system consumes the most time executing a base or manipulator trajectory. For example, an object is grasped after two NBV scans within 13 minutes, three NBV scans within 16 minutes, and four NBV scans within 19 minutes. Although each additional scan increases total completion time by approximately 3 minutes, approximately 20 seconds is consumed completing all predictions to reach an NBV; the remaining time is consumed by real-time motion execution. Within 20 seconds, an NBV can be predicted and reached by a manipulator for object modelling.

Computationally, Pick Planning time is the second-most expensive task and required over three minutes to process all IK for fifty reservoir samples at each base pose. In Table 4.5, short Pick Planning time is observed, and this is due to these trials having approximately three times fewer grasp pose candidates other trials (i.e. 5-6 vs 18-24 solutions). Fewer grasp pose candidates and reservoir samples reduce Pick Planning time, but this causes the system to replan more frequently for another grasp. Replanning increases the Total Run Time, i.e. robot motion is the most time consuming task, because the mobile manipulator needs to move to a new base locations. For all experiments, the system reached the desired grasp pose within 0.5cm, correcting base pose uncertainty ranging from 7.6cm to 15.4cm.

Figures 4.19, 4.20, and 4.21 show the mobile manipulator's system described throughout this work grasping a tin can, box, and cone respectively. Figure 4.19a-d visualizes Clear Room State, described in Section 2.5.1; in an unknown environment, a pre-defined motion moves the manipulator to observe the room. Initially, the system is surrounded by unobserved space and cannot move until free space is detected within the environment. In Figure 4.19e, base poses are sampled and NBVs are projected back to the mobile manipulator to determine if they are reachable; this process is described in Section 2.9.1. Once a prediction is discovered, Navigation State moves the mobile manipulator to a base pose in Figure 4.19f, and Figure 4.19f-I shows Model State modelling the unknown object with a partial point cloud, i.e. Section 3.4.1. Grasp pose generation, i.e. Section 4.4, detected lateral grasps from the partial point cloud model. Pick State activates during this event, and Figure 4.19j shows the system using reservoir sampling, i.e. Section 3.6.1, to predict and rank base pick pose candidates that complete

a Cartesian trajectory to reach the final grasp pose, i.e. Section 2.9.2. Once a base candidate is selected, the mobile system moves near that location and performs pose correction, discussed in Section 3.6, to correct base pose uncertainty. Once corrected, the true final pick locations, shown in Figure 4.19m, are determined to be much further away than predicted, but due to prediction from reservoir sampling, a complete final pick trajectory is discovered and executed to lift the object, shown in Figure 4.19o. Behaviour observed for the cylinder and wooden box is similar. At each experiment's end, the mobile manipulator grasped, lifted, and held each object; these results are shown in Figure 4.22.

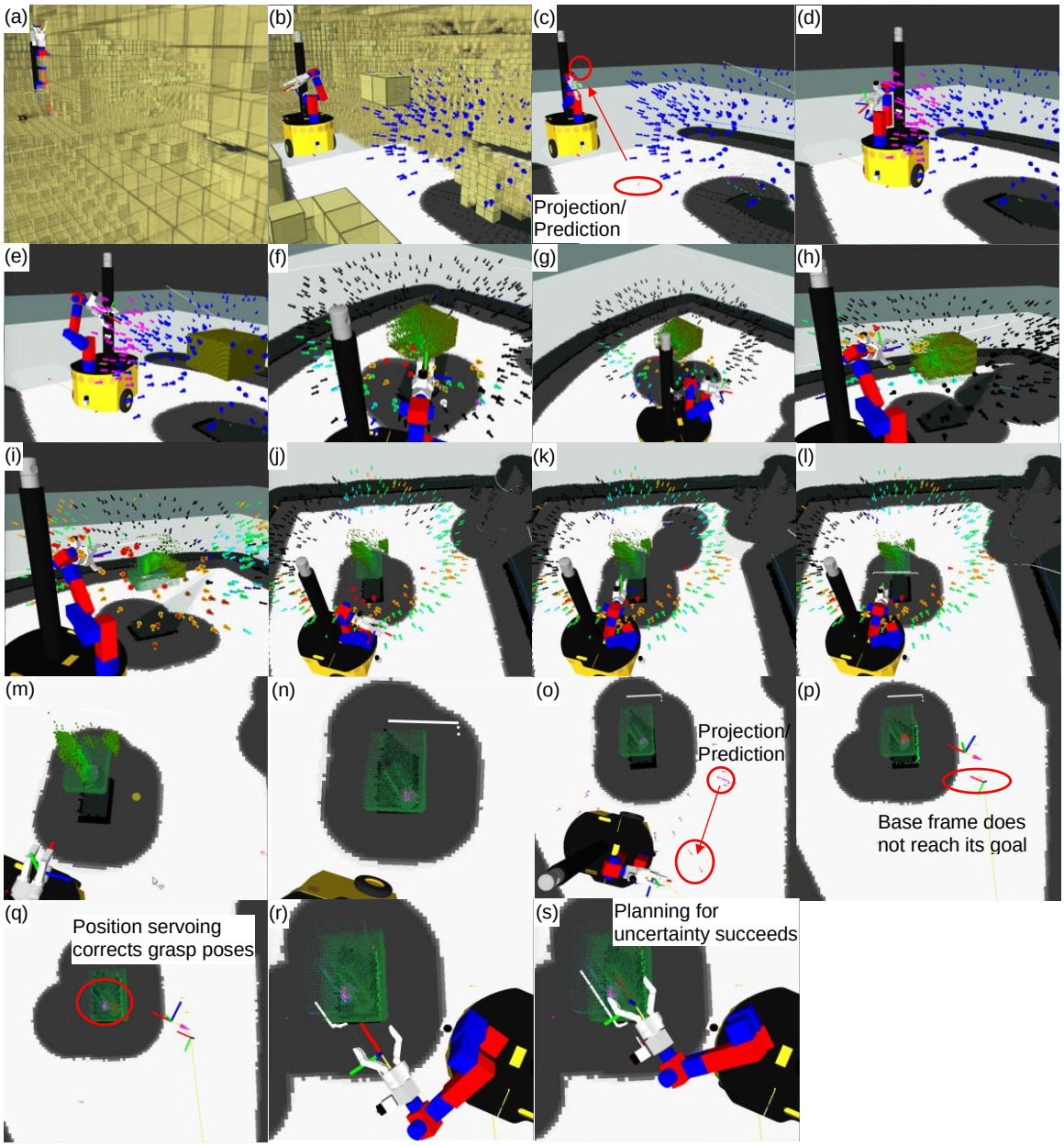


Figure 4.20: Experimental Results Autonomously Grasping an Unknown Box Object†
† Lateral Grasp (magenta), Warmer coloured arrows have the highest ranks,
Yellow voxels are unknown (or unobserved) regions to avoid, Coloured Gripper Axis is Grasp Frame

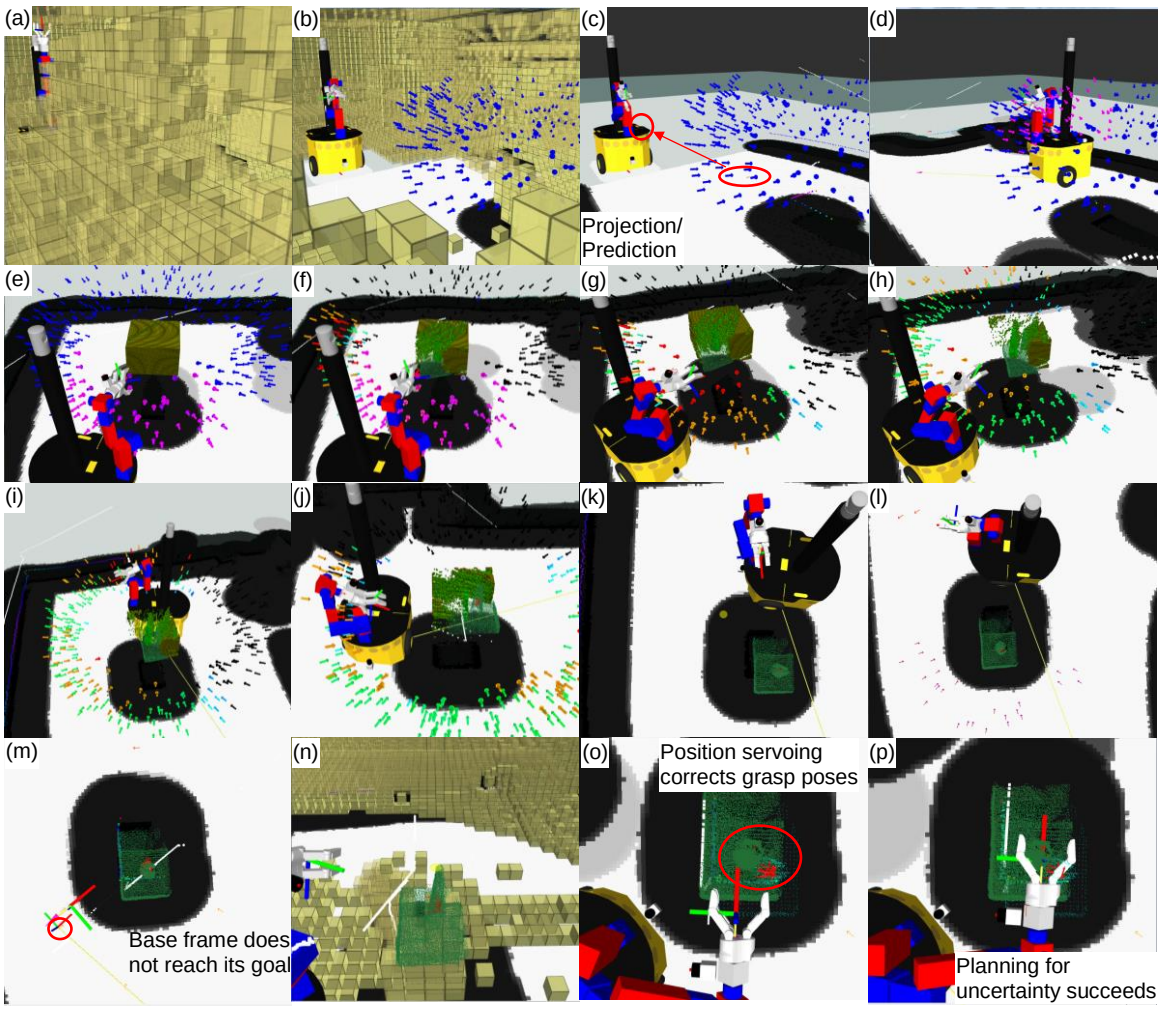


Figure 4.21: Experimental Results Autonomously Grasping an Unknown Cone Object†
† Power Grasp (red), Warmer coloured arrows have the highest ranks,
Yellow voxels are unknown (or unobserved) regions to avoid, Coloured Gripper Axis is Grasp Frame

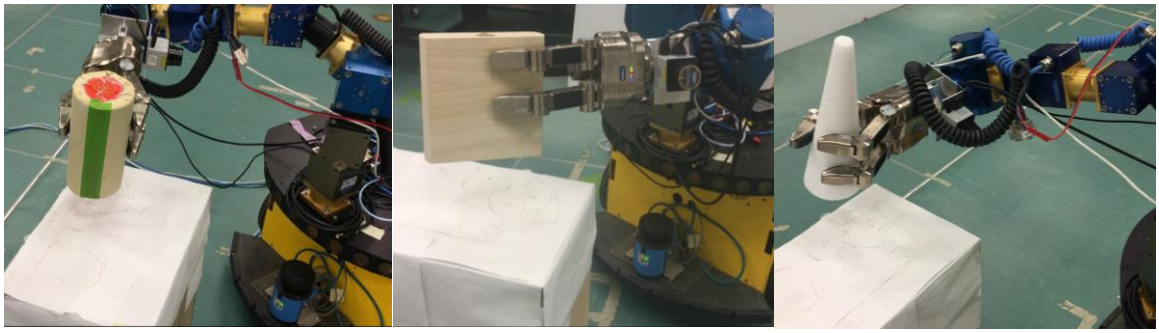


Figure 4.22: Experimental Results Autonomously Lifting and Holding Objects

Chapter 5.

Conclusions and Future Work

5.1. Conclusions

In this thesis, an integrated autonomous grasping system designed for a mobile manipulator is presented. The system is able to plan and execute grasps for a priori unknown objects in unknown environments. Our approach integrates collision-free motion planning and a next best view algorithm to collect and register multiple object scans in a point cloud model. From this point cloud model, a novel multiple grasp type generation algorithm is proposed to place different gripper configurations around the object for grasping.

The grasp generation algorithm identifies grasp locations for multiple grasp types for varying object sizes in near real time. Two key ideas behind the algorithm are: 1) a surface normal histogram guides a gripper orientation search, and 2) voxel grid representation of a gripper and object is cross-correlated to discover a grasp pose for multiple grasp types. Gripper models for cross-correlation are generalized to find grasps for objects of different widths and shapes. Lastly, task-based grasping is presented that utilizes grasp overlap as a feature to identify locations to complete either robot-to-human transfer or securing the object tasks. Voxel size variation shows grasp results remain consistent for different resolutions.

At a system level, large base pose uncertainty is mitigated to complete a precise grasping task. Our system is fully automated, and modelling and grasping an object is successfully achieved despite base uncertainty. An NBV algorithm is simplified to prioritize scan overlap as a key feature to reduce modelling error and assist registration from different viewpoints. Improving registration correspondence by concatenating a 3D point with its respective 3D normal, improves point cloud model reconstruction. Reservoir sampling grasp poses near a region mitigates uncertainty by discovering base pose candidates that yield many IK to a final grasp goal.

5.2. Future Work

While we have shown preliminary experiments executing resultant grasps, in future, we plan to demonstrate our system grasping and lifting an object for different tasks as benchmarking protocols stated in [93]. Preferably, these demonstrations are performed on a fixed-base manipulator using a 2D eye-in-hand lidar to model faster without base pose uncertainty affecting the final grasp pose. A fixed-base manipulator also permits more demonstrations and repeatable tests.

Automatically determining voxel size resolution can be explored further. Results show small and large voxel size are more appropriate to discover grasp pose solutions from small (e.g.. a banana) and large objects (e.g. a bottle) respectively. A complete grasp planning algorithm would need to incorporate several resolutions (or octaves) to accommodate grasping a larger object variety. We show grasp pose generation is consistent within neighbouring resolutions. As a result, further exploration can be conducted to automatically select resolution using techniques from scale invariant literature [94-98].

Incorporating more contextual information would improve grasping performance. For example, an additional stage can be added when observing multiple objects to determine a grasp order. Our generalized algorithm generated an interesting result for multiple objects, i.e. grasp from the outside away from a close object, but our system has no preference to select the drill or tin can first. Context (or preference) needs further exploration along with obstacle avoidance. A straight line path works because most surfaces grasped are smooth. If a ring grasp type (i.e. threading a finger through a mug's ring or placing a gripper inside a handle) is selected, a more appropriate final path needs to be defined to accommodate these grasp types.

References

[1] C. A. Stanger, C. Anglin, W. S. Harwin, and D. P. Romilly, "Devices for assisting manipulation: a summary of user task priorities," *IEEE Trans. on Rehabilitation Engineering,* vol. 2, pp. 256-265, 1994.

[2] L. Torabi and K. Gupta, "Integrated view and path planning for an autonomous six-DOF eye-in-hand object modeling system," in *2010 IEEE/RSJ Int.l Conf. on Intelligent Robots and Systems (IROS)*, 2010, pp. 4516-4521.

[3] V. Pilania and K. Gupta, "Mobile manipulator planning under uncertainty in unknown environments," *The Int. J. of Robotics Research (IJRR),* vol. 37, pp. 316-339, 2018.

[4] C. Eppner, S. Höfer, R. Jonschkowski, R. Martín-Martín, A. Sieverling, V. Wall*, et al.*, "Lessons from the Amazon Picking Challenge: Four aspects of building robotic systems," in *Robotics: Science and Systems*, 2016.

[5] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza*, et al.*, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," presented at the IEEE Int. Conf on Robotics and Automation (ICRA), 2018.

[6] K. Yamazaki, M. Tomono, T. Tsubouchi, and S.-i. Yuta, "A grasp planning for picking up an unknown object for a mobile manipulator," in *2006 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2006, pp. 2143-2149.

[7] S. Jain and B. Argall, "Grasp detection for assistive robotic manipulation," in *2016 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2016, pp. 2015-2021.

[8] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis—a survey," *IEEE Trans. Robot.,* vol. 30, pp. 289-309, 2014.

[9] Q. Lei, J. Meijer, and M. Wisse, "A survey of unknown object grasping and our fast grasping algorithm-C shape grasping," in *2017 3rd Int. Conf. on Control Automation and Robotics*, 2017, pp. 150-157.

[10] B. Wang, L. Jiang, J. Li, and H. Cai, "Grasping unknown objects based on 3d model reconstruction," in *2005 IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics (AIM)*, 2005, pp. 461-466.

[11] G. M. Bone, A. Lambert, and M. Edwards, "Automated modeling and robotic grasping of unknown three-dimensional objects," in *2008 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2008, pp. 292-298.

[12] I. Gori, U. Pattacini, V. Tikhanoff, and G. Metta, "Ranking the good points: A comprehensive method for humanoid robots to grasp unknown objects," in *16th Int. Conf. on Advanced Robotics (ICAR)*, 2013, pp. 1-7.

[13] I. Gori, U. Pattacini, V. Tikhanoff, and G. Metta, "Three-finger precision grasp on incomplete 3d point clouds," in *2014 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2014, pp. 5366-5373.

[14] Q. Lei and M. Wisse, "Fast grasping of unknown objects using force balance optimization," in *2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014, pp. 2454-2460.

[15] Q. Lei and M. Wisse, "Unknown object grasping using force balance exploration on a partial point cloud," in *2015 IEEE Int. Conf. on Advanced Intelligent Mechatronics (AIM)*, 2015, pp. 7-14.

[16] Q. Lei and M. Wisse, "Fast grasping of unknown objects using cylinder searching on a single point cloud," in *9th Int. Conf. on Machine Vision (ICMV)*, 2016, p. 1034108.

[17] T. Suzuki and T. Oka, "Grasping of unknown objects on a planar surface using a single depth image," in *2016 IEEE Int. Conf. on Advanced Intelligent Mechatronics (AIM)*, 2016, pp. 572-577.

[18] C. Connolly, "The determination of next best views," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 1985, pp. 432-435.

[19] R. Pito, "A solution to the next best view problem for automated surface acquisition," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 21, pp. 1016-1030, 1999.

[20] R. Fisher and J. Sanchiz, "A next-best-view algorithm for 3d scene recovery with 5 degrees of freedom," in *British Machine Vision Conf. (BMVC)*, 1999.

[21] W. R. Scott, G. Roth, and J.-F. Rivest, "View planning with a registration constraint," in *Proceedings Third Int. Conf. on 3-D Digital Imaging and Modeling*, 2001, pp. 127-134.

[22] M. Karaszewski, M. Adamczyk, and R. Sitnik, "Assessment of next-best-view algorithms performance with various 3D scanners and manipulator," *J. of Photogrammetry and Remote Sensing (ISPRS),* vol. 119, pp. 320-333, 2016.

[23] S. Kriegel, C. Rink, T. Bodenmüller, and M. Suppa, "Efficient next-best-scan planning for autonomous 3d surface reconstruction of unknown objects," *J. of Real-Time Image Processing,* vol. 10, pp. 611-631, 2015.

[24] C. Maniatis, M. Saval-Calvo, R. Tylecek, and R. B. Fisher, "Best viewpoint tracking for camera mounted on robotic arm with dynamic obstacles," in *2017 Int. Conf. on 3D Vision (3DV)*, 2017, pp. 107-115.

[25] J. I. Vasquez-Gomez, L. E. Sucar, R. Murrieta-Cid, and E. Lopez-Damian, "Volumetric next-best-view planning for 3D object reconstruction with positioning error," *Int. J. of Advanced Robotic Systems,* vol. 11, p. 159, 2014.

[26] J. I. Vasquez-Gomez, L. E. Sucar, and R. Murrieta-Cid, "View/state planning for three-dimensional object reconstruction under uncertainty," *Autonomous Robots,* vol. 41, pp. 89-109, 2017.

[27] J. I. Vasquez-Gomez, L. E. Sucar, R. Murrieta-Cid, and J.-C. Herrera-Lozada, "Tree-based search of the next best view/state for three-dimensional object reconstruction," *Int. J. of Advanced Robotic Systems,* vol. 15, p. 1729881418754575, 2018.

[28] J. Daudelin and M. Campbell, "An adaptable, probabilistic, next-best view algorithm for reconstruction of unknown 3-d objects," *IEEE Robotics and Automation Letters,* vol. 2, pp. 1540-1547, 2017.

[29] J.-W. Li, H. Liu, and H.-G. Cai, "On computing three-finger force-closure grasps of 2-D and 3-D objects," *IEEE Trans. on Robotics and Automation,* vol. 19, pp. 155-161, 2003.

[30] X. Zhu and H. Ding, "Planning force-closure grasps on 3-D objects," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2004, pp. 1258-1263.

[31] B. Bounab, D. Sidobre, and A. Zaatri, "Central axis approach for computing n-finger force-closure grasps," in *2008 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2008, pp. 1169-1174.

[32] S. El-Khoury and A. Sahbani, "On computing robust n-finger force-closure grasps of 3D objects," in *2009 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2009, pp. 2480-2486.

[33] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, "Automatic grasp planning using shape primitives," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2003, pp. 1824-1829.

[34] K. Huebner and D. Kragic, "Selection of robot pre-grasps using box-based shape approximation," in *2008 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2008, pp. 1765-1770.

[35] C. Goldfeder, P. K. Allen, C. Lackner, and R. Pelossof, "Grasp planning via decomposition trees," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2007, pp. 4679-4684.

[36] M. Przybylski, T. Asfour, and R. Dillmann, "Planning grasps for robotic hands using a novel object representation based on the medial axis transform," in *2011 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2011, pp. 1781-1788.

[37] M. Nieuwenhuisen, J. Stückler, A. Berner, R. Klein, and S. Behnke, "Shape-primitive based object recognition and grasping," in *7th German Conference on Robotics (ROBOTIK)*, 2012, pp. 1-5.

[38] A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, J. Bohg, T. Asfour*, et al.*, "Learning of grasp selection based on shape-templates," *Autonomous Robots,* vol. 36, pp. 51-65, 2014.

[39] D. Fischinger, A. Weiss, and M. Vincze, "Learning grasps with topographic features," *The Int. J. of Robotics Research (IJRR),* vol. 34, pp. 1167-1194, 2015.

[40] E. Dessalene, Y. H. Ong, J. Morrow, R. Balasubramanian, and C. Grimm, "Using geometric features to represent near-contact behavior in robotic grasping," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2019, pp. 2772-2777.

[41] R. Detry, C. H. Ek, M. Madry, J. Piater, and D. Kragic, "Generalizing grasps across partly similar objects," in *2012 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012, pp. 3791-3797.

[42] L. Berscheid, T. Rühr, and T. Kröger, "Improving data efficiency of self-supervised learning for robotic grasping," in *2019 Int. Conf. on Robotics and Automation (ICRA)*, 2019, pp. 2125-2131.

[43] Q. Lu and T. Hermans, "Modeling grasp type improves learning-based grasp planning," *IEEE Robot. Autom. Letters,* vol. 4, pp. 784-791, 2019.

[44] J. Cai, H. Cheng, Z. Zhang, and J. Su, "MetaGrasp: Data efficient grasping by affordance interpreter network," in *Int. Conf. on Robotics and Automation (ICRA)*, 2019, pp. 4960-4966.

[45] Q. V. Le, D. Kamm, A. F. Kara, and A. Y. Ng, "Learning to grasp objects with multiple contact points," in *2010 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2010, pp. 5062-5069.

[46] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp pose detection in point clouds," *The Int. J. of Robotics Research (IJRR),* vol. 36, pp. 1455-1473, 2017.

[47] H. Liang, X. Ma, S. Li, M. Görner, S. Tang, B. Fang*, et al.*, "Pointnetgpd: Detecting grasp configurations from point sets," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2019, pp. 3629-3635.

[48] H. Karaoguz and P. Jensfelt, "Object detection approach for robot grasp detection," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2019, pp. 4953-4959.

[49] S. Hasegawa, K. Wada, S. Kitagawa, Y. Uchimi, K. Okada, and M. Inaba, "GraspFusion: Realizing complex motion by learning and fusing grasp modalities with instance segmentation," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2019, pp. 7235-7241.

[50] H. B. Amor, O. Kroemer, U. Hillenbrand, G. Neumann, and J. Peters, "Generalization of human grasping for multi-fingered robot hands," in *2012 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012, pp. 2043-2050.

[51] C. Goldfeder, M. Ciocarlie, H. Dang, and P. K. Allen, "The columbia grasp database," in *2009 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2009, pp. 1710-1716.

[52] M. J. Hegedus, K. Gupta, and M. Mehrandezh, "Towards an integrated autonomous data-driven grasping system with a mobile manipulator," *IEEE Int. Conf. on Robotics and Automation (ICRA),* pp. 1596-1600, May 20-24, 2019.

[53] M. J. Davari, M. J. Hegedus, K. Gupta, and M. Mehrandezh, "Identifying multiple interaction events from tactile data during robot-human object transfer," *28th IEEE Int. Conf. on Robot and Human Interactive Communication (RO-MAN),* pp. 1-6, Oct. 14-18, 2019.

[54] G. Grisetti, G. D. Tipaldi, C. Stachniss, W. Burgard, and D. Nardi, "Fast and accurate SLAM with Rao–Blackwellized particle filters," *Robotics and Autonomous Systems,* vol. 55, pp. 30-38, 2007.

[55] R. S. Hartenberg and J. Denavit, "A kinematic notation for lower pair mechanisms based on matrices," *J. of Applied Mechanics,* vol. 77, pp. 215-221, 1955.

[56] D. Meagher, "Geometric modeling using octree encoding," *Computer graphics and image processing,* vol. 19, pp. 129-147, 1982.

[57] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots,* vol. 34, pp. 189-206, 2013.

[58] A. Williams, S. Barrus, R. K. Morley, and P. Shirley, "An efficient and robust ray-box intersection algorithm," in *ACM SIGGRAPH 2005 Courses*, 2005, pp. 1-4.

[59] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *2010 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2010, pp. 300-307.

[60] S. Oajsalee, S. Tantrairatn, and S. Khaengkarn, "Study of ROS based localization and mapping for closed area survey," in *IEEE 5th Int. Conf. on Mechatronics System and Robots (ICMSR)*, 2019, pp. 24-28.

[61] S. Chitta, "Moveit!: an introduction," in *Robot Operating System (ROS)*, ed: Springer, 2016, pp. 3-27.

[62] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ros topics]," *IEEE Robotics & Automation Magazine,* vol. 19, pp. 18-19, 2012.

[63] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," in *Sensor Fusion IV: Control Paradigms and Data Structures*, 1992, pp. 586-607.

[64] W. R. Scott, G. Roth, and J.-F. Rivest, "View planning for automated three-dimensional object reconstruction and inspection," *ACM Computing Surveys (CSUR),* vol. 35, pp. 64-96, 2003.

[65] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2000, pp. 995-1001.

[66] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp," in *Robotics: Science and Systems*, 2009, p. 435.

[67] J. Yang, H. Li, D. Campbell, and Y. Jia, "Go-ICP: A globally optimal solution to 3D ICP point-set registration," in *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2016, pp. 2241-2254.

[68] Y. He, B. Liang, J. Yang, S. Li, and J. He, "An iterative closest points algorithm for registration of 3D laser scanner point clouds with geometric features," *Sensors,* vol. 17, p. 1862, 2017.

[69] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments," *The Int. J. of Robotics Research (IJRR),* vol. 31, pp. 647-663, 2012.

[70] J. Weisz and P. K. Allen, "Pose error robust grasping from contact wrench space metrics," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012, pp. 557-562.

[71] D. Seita, F. T. Pokorny, J. Mahler, D. Kragic, M. Franklin, J. Canny*, et al.*, "Large-scale supervised learning of the grasp robustness of surface patch pairs," in *IEEE Int. Conf. on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, 2016, pp. 216-223.

[72] J. S. Vitter, "Random sampling with a reservoir," *ACM Transactions on Mathematical Software (TOMS),* vol. 11, pp. 37-57, 1985.

[73] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine,* vol. 19, pp. 72-82, 2012.

[74] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine,* vol. 4, pp. 23-33, 1997.

[75] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (PCL)," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 1-4.

[76] R. Balasubramanian, L. Xu, P. D. Brook, J. R. Smith, and Y. Matsuoka, "Physical human interactive guidance: Identifying grasping principles from human-planned grasps," *IEEE Tran. on Robotics,* vol. 28, pp. 899-910, 2012.

[77] V.-D. Nguyen, "Constructing force-closure grasps," *The Int. J. of Robotics Research (IJRR),* vol. 7, pp. 3-16, 1988.

[78] M. R. Cutkosky, "On grasp choice, grasp models, and the design of hands for manufacturing tasks," *IEEE Trans. on Robotics and Automation,* vol. 5, pp. 269-279, 1989.

[79] E. Rimon and A. Blake, "Caging 2D bodies by 1-parameter two-fingered gripping systems," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 1996, pp. 1458-1464.

[80] A. S. Besicovitch, "A net to hold a sphere," *The Mathematical Gazette,* vol. 41, pp. 106-107, 1957.

[81] D. Seita, F. T. Pokorny, J. Mahler, D. Kragic, M. Franklin, J. Canny*, et al.*, "Large-scale supervised learning of the grasp robustness of surface patch pairs," in *2016 IEEE Int. Conf. on Simul., Model., and Program. for Auton. Robots (SIMPAR)*, 2016, pp. 216-223.

[82] F. Heinemann, S. Puhlmann, C. Eppner, J. Élvarez-Ruiz, M. Maertens, and O. Brock, "A taxonomy of human grasping behavior suitable for transfer to robotic hands," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2015, pp. 4286-4291.

[83] T. Feix, J. Romero, H.-B. Schmiedmayer, A. M. Dollar, and D. Kragic, "The grasp taxonomy of human grasp types," *IEEE Trans. on Human-Machine Systems,* vol. 46, pp. 66-77, 2015.

[84] N. Rojas and A. M. Dollar, "Classification and kinematic equivalents of contact types for fingertip-based robot hand manipulation," *J. of Mechanisms and Robotics,* vol. 8, 2016.

[85] F. Stival, S. Michieletto, M. Cognolato, E. Pagello, H. Müller, and M. Atzori, "A quantitative taxonomy of human hand grasps," *J. of NeuroEngineering and Rehabilitation,* vol. 16, pp. 1-17, 2019.

[86] L. E. Kavraki, "Computation of configuration-space obstacles using the fast fourier transform," *IEEE Trans. on Robotics and Automation,* vol. 11, pp. 408-413, 1995.

[87] V. Bhatia Nitin, "Survey of nearest neighbor techniques," *Int. J. of Computer Science and Information Security,* vol. 2, pp. 302-305, 2010.

[88] R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for point-cloud shape detection," in *Computer graphics forum*, 2007, pp. 214-226.

[89] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM,* vol. 24, pp. 381-395, 1981.

[90] M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," *Proceedings of the IEEE,* vol. 93, pp. 216-231, 2005.

[91] H. Dang and P. K. Allen, "Stable grasping under pose uncertainty using tactile feedback," *Auton. Robots,* vol. 36, pp. 309-330, 2014.

[92] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The ycb object and model set: Towards common benchmarks for manipulation research," in *2015 Int. Conf. on Advanced Robotics (ICAR)*, 2015, pp. 510-517.

[93] Y. Bekiroglu, N. Marturi, M. A. Roa, K. J. M. Adjigble, T. Pardi, C. Grimm*, et al.*, "Benchmarking protocol for grasp planning algorithms," *IEEE Robot. and Autom. Lett.,* vol. 5, pp. 315-322, 2019.

[94] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. of Computer Vision,* vol. 60, pp. 91-110, 2004.

[95] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*, 2006, pp. 404-417.

[96] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European conference on computer vision*, 2006, pp. 430-443.

[97] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *European Conf. on Computer Vision*, 2010, pp. 778-792.

[98] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 Int. Conf. on Computer Vision*, 2011, pp. 2564-2571.

[99] X. Markenscoff, L. Ni, and C. H. Papadimitriou, "The geometry of grasping," *The Int. J. of Robotics Research (IJRR),* vol. 9, pp. 61-74, 1990.

[100] A. Bicchi, "On the closure properties of robotic grasping," *The Int. J. of Robotics Research (IJRR),* vol. 14, pp. 319-334, 1995.

[101] T. Rabbani, F. Van Den Heuvel, and G. Vosselmann, "Segmentation of point clouds using smoothness constraint," *Int. Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS),* vol. 36, pp. 248-253, 2006.

Appendix A.

Using Finger Patches for Grasping

Grasp Planning Phase with Uncertainty

As the partial point cloud model is updated with each current i^{th} scan S_i, grasp analysis is conducted for a grasp. Our goal is to generate finger sized patches within the current point cloud I_i to perform force closure analysis[99, 100]. The cloud's surface is represented with finger-tip sized patches to reduce computational load, mitigate noise within the point cloud model, and identify regions to place a finger. Patches along the object's surface is shown in Figure A1.

Identifying Finger-Sized Contact Patches

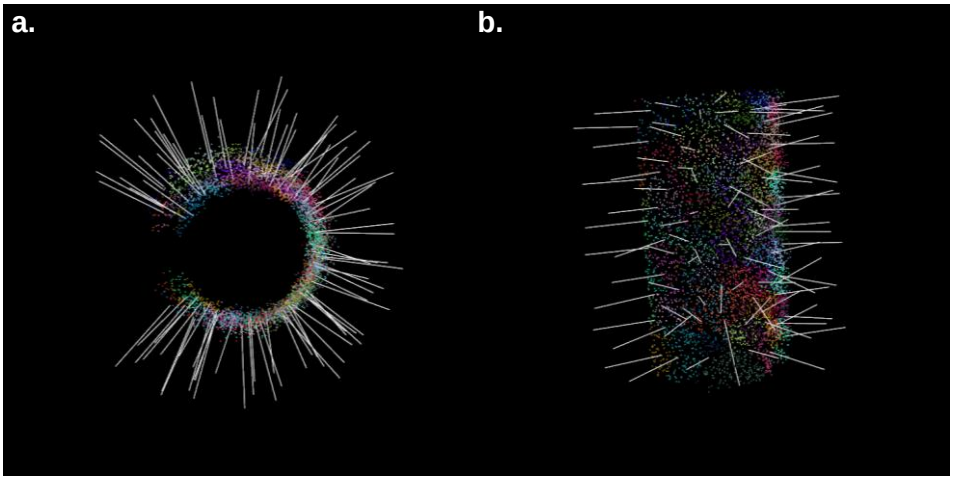


Figure A1. Top & Side View of a Cylinder Represented with Patches

The incoming point cloud I_i is initially downsampled to uniformly distribute points and remove high frequency noise. The center of mass (CoM) is estimated by using the mean of all points in I_i. Without downsampling, any further estimates for CoM biases towards a region scanned frequently instead of approaching the true object's CoM. High-frequency noise is smoothed because it affects normal direction estimates. In addition, the faces of I_i are segmented with region growing segmentation using local curvature[101]; object edges are removed by excluding segments with a large curvature which improves normal direction estimates and biases grasp selection along the object faces. Small

faces are removed if their surface is smaller than our gripper's finger. Surface faces are desired as they are more stable than edges to grasp. Finger-sized patches are generated by randomly selecting a point $\mathbf{P}_{\mathbf{rnd}}$ within the remaining faces of I_i; any points within a user defined local radius (2.5cm in our case) of $\mathbf{P}_{\mathbf{rnd}}$ are discovered, their corresponding normals are averaged to create $\mathbf{N}_{\mathbf{rnd}}$, and both $\mathbf{P}_{\mathbf{rnd}}$ & $\mathbf{N}_{\mathbf{rnd}}$ are inserted into finger-sized patch list **F** to test force closure. All locally selected points are flagged to not be re-selected by as $\mathbf{P}_{\mathbf{rnd}}$.

Grasp Analysis for Finger-Sized Patches

To complete force closure analysis for any set of contact points from fingers, a grasp wrench space (GWS) is created. A GWS is a six-dimensional space that represents all forces and torques that can be applied to the OI [99, 100]. To estimate the GWS, the friction coefficient μ, center of mass (CoM), applied force, contact location, and surface normal at each contact need to be known. Any large uncertainties associated with these parameters may cause a failed grasp, but this can be addressed. If the CoM location is uncertain, the object may twist and slip. In addition, the OI can also slip if its friction coefficient is very small, but practically, the gripper can apply a stronger force to mitigate these problems. Surface normals estimated contain uncertainty that affect a friction cone's orientation, but this is lessened by averaging a patch of surface normals.

As no information is given about the object, we assume a friction coefficient μ=0.8 to represent rubber contacts; a hypercube is selected for a task wrench space (TWS) to represent forces and torques applied in all directions. All other parameters are empirically derived from the point cloud model. If a GWS contains the TWS about its origin, force closure is deemed satisfied, and the contact pair is ranked based on the TWS volume. The final grasp pose, written as G_f, is calculated by averaging two contact locations on the object's surface. Orientation of G_f is relative to two surface patch normals, n_1 and n_2. The x-axis of the grasp frame is created by adding two inverted contact normals. Simple kinematics determines the remaining axes.

Appendix B.

ROS Node Mobile Manipulator Settings

Base Navigation Costmap Parameters

map_type: costmap

transform_tolerance: 3.0

obstacle_range: 4.5

max_obstacle_height: 2.0

raytrace_range: 4.0

#Powerbot's footprint 0.84 x 0.63

footprint: [[-0.385,-0.350], [-0.545,-0.280], [-0.545,0.280], [-0.385,0.350], [0.205,0.350], [0.370,0.280], [0.370,-0.280], [0.205,-0.350]]

inflation_radius: 0.385

cost_scaling_factor: 8.3

lethal_cost_threshold: 105

observation_sources: base_scan

base_scan: {sensor_frame: base_scan_link, topic: /scan_filtered, data_type: LaserScan, expected_update_rate: 5.0, observation_persistance: 0.0, marking: true, clearing: true}

global_costmap:

- global_frame: /map
- robot_base_frame: /base_link
- update_frequency: 5.0
- publish_frequency: 2.0
- raytrace_range: 30.0
- obstacle_range: 18
- static_map: true
- rolling_window: false
- width: 24.0
- height: 24.0
- resolution: 0.025

local_costmap:

- global_frame: /odom
- robot_base_frame: /base_link
- update_frequency: 5.0
- publish_frequency: 2.0
- static_map: false
- rolling_window: true
- width: 12.0
- height: 12.0
- resolution: 0.025
- origin_x: 0.0
- origin_y: 0.0

origin_x: -12.0

origin_y: -12.0

track_unknown_space: true

unknown_cost_value: 255

DWA Planner Parameters

base_local_planner: dwa_local_planner/DWAPlannerROS

--- recovery behaviours ---

recovery_behaviors: [{name: conservative_reset, type: clear_costmap_recovery/ClearCostmapRecovery}, {name: rotate_recovery, type: rotate_recovery/RotateRecovery}, {name: aggressive_reset, type: clear_costmap_recovery/ClearCostmapRecovery}]

conservative_reset_dist: 2.5

recovery_behavior_enabled: true

clearing_rotation_allowed: true

DWAPlannerROS:

transform_tolerance: 3.0

world_model: costmap

#Goal Tolerance Settings

xy_goal_tolerance: 0.12

yaw_goal_tolerance: 0.0960

latch_xy_goal_tolerance: true

#Robot Configuration

acc_lim_x: 7.0

acc_lim_y: 0.0

acc_lim_theta: 4.0

max_vel_x: 0.30

min_vel_x: -0.16

max_vel_y: 0

min_vel_y: 0

max_trans_vel: 0.7

min_trans_vel: 0.05

max_rot_vel: 1.047

#Robot Configuration

acc_lim_x: 7.0

acc_lim_y: 0.0

acc_lim_theta: 4.0

max_vel_x: 0.30

min_vel_x: -0.16

max_vel_y: 0

min_vel_y: 0

max_trans_vel: 0.7

min_trans_vel: 0.05

max_rot_vel: 1.047

min_rot_vel: 0

escape_vel: -0.1

holonomic_robot: false

#Base Local Planner Configs

dwa: true

meter_scoring: true

simple_attractor: false

```
        min_rot_vel: 0
        escape_vel: -0.1
        holonomic_robot: false

 #Weights to affect planning (dwa_planner)
        goal_distance_bias: 20
        path_distance_bias: 30
        occdist_scale: 0.06
        scaling_speed: 0.25
        max_scaling_factor: 0.2
        forward_point_distance: 0.325
        oscillation_reset_dist: 0.025
        stop_time_buffer: 0.2
        prune_plan: true
```

```
        use_dwa: false
```

World Octomap Server

```
<node pkg="octomap_server" type="octomap_server_node" name="coarse_octomap_server">
        <param name="publish_free_space" value="false" />
        <param name="resolution" value="0.1" />
        <param name="frame_id" type="string" value="/map" />
        <param name="max_sensor_range" value="8.0" />
        <param name="sensor_model/max_range" value = "7.5"/>
        <param name="sensor_model/hit" value = "0.7"/>
        <param name="sensor_model/miss" value = "0.07"/>
        <param name="sensor_model/max" value = "0.97"/>
        <param name="sensor_model/min" value = "0.25"/>

        <param name="latch" value="false" />
        <param name="filter_ground " value="false" />
        <param name="filter_speckles" value="true" />
</node>
```

Model Octomap Server

```
<node pkg="octomap_server" type="octomap_server_node" name="fine_octomap_server" >
```

```
        <param name="publish_free_space" value="false" />
        <param name="frame_id" type="string" value="/map" />
        <param name="resolution" value="0.008" />
        <param name="sensor_model/max_range" value = "5.0"/>
        <param name="sensor_model/hit" value = "0.75"/>
        <param name="sensor_model/miss" value = "0.35"/>
        <param name="sensor_model/max" value = "0.98"/>
        <param name="sensor_model/min" value = "0.12"/>

        <param name="latch" value="false" />
        <param name="filter_speckles" value="true" />
        <param name="filter_ground" value="false" />
        <param name="publish_free_space" value="true" />
</node>
```

Point Cloud Registration

```
#Implements PCL's v1.7 Iterative Closest Point Library
<node type="pointcloud_assembler" pkg="pcl_processing" name="pc_assembler"
output="screen">
        <param name="en_control" type="bool" value="true" />
<!--Voxel Filtering/Downsampling Parameters Input Cloud-->
        <param name="leaf_x" type="double" value="0.0040" />
        <param name="leaf_y" type="double" value="0.0040" />
        <param name="leaf_z" type="double" value="0.0040" />

<!--Statistics Removals Parameters- Input Cloud->
        <param name="meank" type="double" value="12" />
        <param name="std_thresh" type="double" value="2.0" />

<!--GICP Parameters-->
        <param name="max_iterations" type="int" value="200" />
        <param name="euclidean_fitness_epsilon" type="double" value="0.001" />
        <param name="transformation_epsilon" type="double" value="1e-10" />
        <param name="max_correspondence_distance" type="double" value="0.125" />
        <param name="min_cloud_size" type="int" value="80" />

<!--GICP Scaling for point representation (x,y,z, nx,ny,nz, curvature)-->
```

```
        <param name="alpha_x" type="double" value="1.0" />
        <param name="alpha_y" type="double" value="1.0" />
        <param name="alpha_z" type="double" value="1.0" />
        <param name="alpha_nx" type="double" value="1.0" />
        <param name="alpha_ny" type="double" value="1.0" />
        <param name="alpha_nz" type="double" value="1.0" />
        <param name="alpha_c" type="double" value="1.0" />

<!--GICP Resolution for Merging Pointclouds (downsample factor)-->
        <param name="icp_leaf_x" type="double" value="0.0065" />
        <param name="icp_leaf_y" type="double" value="0.0065" />
        <param name="icp_leaf_z" type="double" value="0.0065" />

<!--Correspondence Rejector Parameters-->
        <param name="RANSAC_max_iterations" type="int" value="100" />
        <param name="RANSAC_outlier_thresh" type="double" value="0.005" />
        <param name="median_correspondance_factor" type="double" value="0.5" />
        <param name="normal_correspondance_angle_deg" type="double" value="20" />
        <param name="min_var_trimmed_ratio" type="double" value="0.05" />
        <param name="max_var_trimmed_ratio" type="double" value="0.95" />
        <param name="min_overlap_ratio" type="double" value="0.03" />
        <param name="exp_overlap_ratio" type="double" value="0.25" />
</node>
```