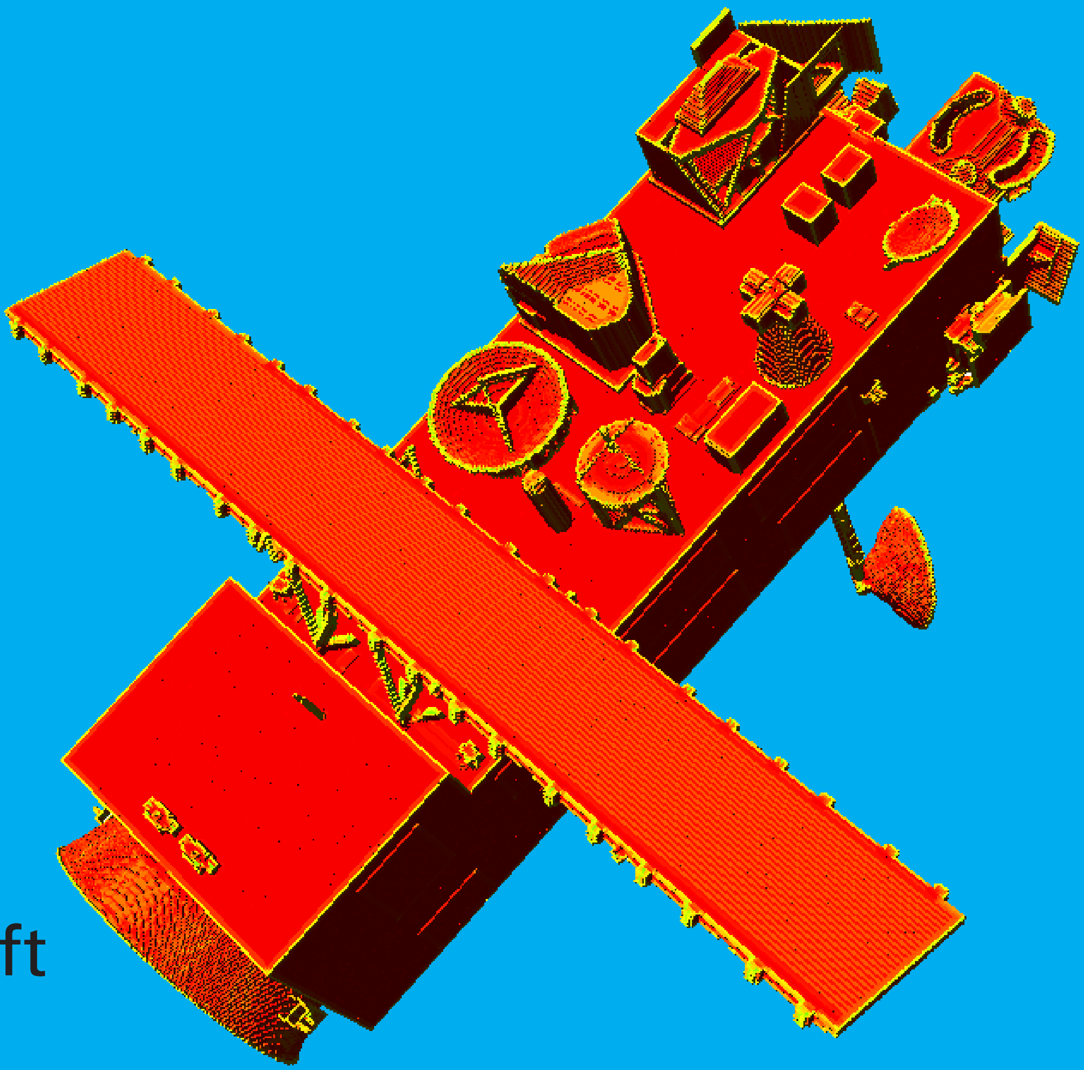
Global Grasp Planning for On-Orbit Robotic Manipulation

Design of an analytic grasp planning engine for autonomous capture of novel objects

M. Vilella Ramisa



TUDelft

Global Grasp Planning for On-Orbit Robotic Manipulation

Design of an analytic grasp planning engine for autonomous capture of novel objects

by

M. Vilella Ramisa

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday August 30, 2021 at 2:00 PM.

Student number:	4447301	
Project duration:	September 1, 2020 — August 13, 2021	
Thesis committee:	Dr. A. Cervone,	TU Delft, chair
	Dr. A. Menicucci,	TU Delft, supervisor
	M. Zwick, M.Eng.,	European Space Agency, supervisor
	Dr. M.J. Heiligers,	TU Delft, examiner

TU Delft

Abstract

To manage the increasing amount of satellite traffic in orbit and renewed operational ambitions, it is crucial to advance the capabilities of space robotic systems to enable complex manipulability tasks. These abilities will provide the necessary components to perform satellite de-orbiting, servicing of important assets, and assistance in on-orbit operations.

With knowledge transfer from the terrestrial domain, the Aut-O-MAGIC research project proposes global graspability maps based on analytic grasp quality metrics to enable robust gripper operations without external intervention. Global graspability maps draw parallels from terrain traversability maps in robotic navigation, which use cost maps to plan optimised traverses ahead of time. Until now, grasping tasks in space have only had sufficient target surface information to plan the next action, disregarding optimisation or complex tasks that require multiple actions. The Aut-O-MAGIC grasp planning engine provides promising graspability cost maps that indicate populous sets of satisfactory grasps on the target object surface.

Preface

This thesis marks the end of my period as a student at TU Delft, which has been a great privilege. I would have never expected to become as integrated in Dutch culture and life as I have, and I will look back at this student period fondly.

I would like to thank my supervisors: Martin Zwick and Alessandra Menicucci. Martin Zwick believed in me, which was evident when he arranged for me to have a second internship opportunity at ESTEC, and I greatly appreciate that. I feel grateful of both Martin and Alessandra, who provided me invaluable guidance and feedback.

I would also like to thank all my colleagues at the ESA Robotics section, TEC-MMA. My time shared with them was very fun and enriching. My experiences there are what made me the engineer I am (about) to become.

Last, but certainly not least, I would like to thank Solenn Walstra and my family for their unwavering support, it would have not been possible without them.

M. Vilella Ramisa
Delft, August 2021

Contents

List of Figures xi

List of Tables xvii

List of Acronyms xix

List of Scripts xxi

I Introduction and Overview 1

1 Introduction 3
- 1.1 Motivation 3
- 1.2 Overview 3
- 1.3 Research Objective and Scope 4

2 Grasp Planning Pipeline Overview 7
- 2.1 Space Proximity Operations 7
  - 2.1.1 Rendezvous and Docking 7
  - 2.1.2 3D Imaging for Relative Navigation 11
  - 2.1.3 Opportunities and Challenges 13
- 2.2 Data Structure 14
  - 2.2.1 Point Cloud 14
  - 2.2.2 Octree 16
- 2.3 Grasp Planning 18
  - 2.3.1 Preprocessing 19
  - 2.3.2 Grasp Quality Metrics 22
  - 2.3.3 Grasp Candidate Generation 28
- 2.4 Improvement Gaps for Space Applications 28

II Framework Design 31

3 Concepts and Architecture 33
- 3.1 Mission Needs and System Environments 34
  - 3.1.1 Need and Mission statement 34
  - 3.1.2 System Environments 34
- 3.2 Stakeholder Requirements 35
  - 3.2.1 The Legacy Context for Validation Tests 35
  - 3.2.2 Stakeholder Requirements 40
- 3.3 Key Performance Parameters and Acceptance Criteria 40
- 3.4 Concept of Operations 42
- 3.5 Conclusions 42

4 System Design **45**
4.1 Technical Requirements Definition 45
4.2 Trade Studies 47
4.2.1 3D Imaging Hardware 47
4.2.2 Software Programming Framework 49
4.3 Hardware Component Selection 51
4.4 Software Component Decomposition 52
4.4.1 System Architecture Model 52
4.4.2 Derived Technical Requirements 52
4.5 Design Solution Definition 55
4.5.1 OctoMap Grasping 55
4.5.2 Grasp Planning Engine 55
4.5.3 3D Model Voxelisation 56
4.5.4 3D Model to Depth Map 58
4.5.5 Scene Reconstruction 58
4.6 Conclusions 59

5 System Implementation **61**
5.1 OctoMap Grasping 61
5.2 Grasp Planning Engine 63
5.3 3D Model to Depth Map 66

III Framework Evaluation **67**

6 System Verification **69**
6.1 Envisat 69
6.2 Dex-Net Simulations 73
6.3 Conclusions 79

7 System Validation **81**
7.1 Experimental Setup 81
7.1.1 The Orbital Robotics and GNC Laboratory and GRALS 82
7.1.2 Payload 84
7.1.3 Target 84
7.2 Trajectory 84
7.3 Data Collection 86
7.4 Graspability Analysis 87
7.5 Conclusions 91

IV Conclusions and Recommendations **95**

8 Requirements Compliance **97**

9 Conclusions **99**

10 Recommendations for Future Work **103**

V Appendices 105

A System Operation 107

A.1 Grasp Planning Engine 107

A.2 3D Model Voxelisation. 109

A.3 3D Model to Depth Map. 111

B Grasp Methods Sensitivity Analysis Visualisations 113

List of Figures

1.1 Mind map illustration of uncooperative Rendezvous and Capture research context . . . 5
1.2 Research framework ‘Autonomous Grasp Planning for Robotic Capture of Space Objects’. . . . 5

2.1 Gemini VIII docking to the Agena target vehicle . . . 8
2.2 MEV-2 IR WFOV camera view at 15m away from Intelsat 10-02 in 2021 8
2.3 Operational range and measurement accuracies of common rendezvous sensors, from Fehse [2]. . . . 11
2.4 Simulation of relative pose between MEV-1 and target Intelsat 901 compared with image taken by IR WFOV proximity camera . . . 13
2.5 K4A colour image. . . . 15
2.6 K4A IR image. . . . 15
2.7 Computed 2D depth image form colour and IR image. . . . 15
2.8 3D depth point cloud projected from depth image depicted in Figure 2.7. 15
2.9 A torus represented by 100, 1k, and 10k voxels. . . . 16
2.10 Gridded voxel representation, from Gebhardt, Payzer, Salemann, *et al.* [25]. . . . 17
2.11 Sparse voxel representation, from Gebhardt, Payzer, Salemann, *et al.* [25]. 17
2.12 Octree voxel representation, from Gebhardt, Payzer, Salemann, *et al.* [25]. 18
2.13 Mind map of the aspects that play a role in the problem of grasp planning, with the aspects within the scope of this thesis work in blue background. . . . 19
2.14 Example 2-D k-Nearest Neighbours analysis . . . 20
2.15 The 15 unique isosurface cases in the Marching Cubes algorithm. . . . 21
2.16 Example grasp assessment ith voxel grid representation, from Hegedus, Gupta, and Mehrandezh [60] . . . 26
2.17 Three form closure grasp proposals with the rightmost not achieving form closure, from Siciliano and Khatib [55] . . . 27
2.18 Examples of grasps generated from four different grasp primitive shapes, from Miller, Knoop, Christensen, *et al.* [34]. . . . 27

3.1 Systems Engineering V-Model for Autonomous On-Orbit Manipulability Analyser for Gripper Interaction Capabilities, emphasising the dual components of decomposition and integration in the product development process. . . . 33
3.2 Workflow on how to run an experiment with the GRALS facility . . . 36
3.3 Simplified topology of the considered systems in GRALS . . . 37
3.4 GRALS hardware connectivity (communications) diagram . . . 38
3.5 Mounting flange taken from the KR10 R1100 sixx C operating instructions 39

3.6 Functional principle of GRALS operation using a remote PC 39
3.7 Maximum mass of payload, depending on its inertial distribution 39
3.8 Figures including the traversability cost maps for rover navigation in different terrains 43
3.9 The ‘as-is’ Environment to be addressed by the Aut-O-MAGIC system, in red are the elements to be replaced. 44
3.10 The envisioned Aut-O-MAGIC system environment, with the green elements substituting the cooperative navigation system block. 44

4.1 Pointclouds in Intel RealSense L515 vs Microsoft Azure Kinect DK . . . 48
4.2 2D depth map and RGB image in Intel RealSense D455 48
4.3 Pugh Matrix for the software communication framework. 50
4.4 The Microsoft Azure Kinect DK time-of-flight camera 51
4.5 Software system architecture model for the Aut-O-MAGIC project. . . . 53
4.6 Gripper object displaying positively-interacting target voxels in solid green, and negatively-interacting target voxels in red. 56
4.7 Gripper object with rays cast from its antipodal plates, as done in methods #2, #4, and #5. 56
4.8 Origin of coordinates of a gripper tree in *loaded* state is at the center of the graspable region. 58

5.1 UML Class Diagram for Octomap-Grasping package. 62
5.2 Dependency graph for `OcTreeGraspQuality.cpp` source file. 63
5.3 Dependency graph for `OcTreeGripper.cpp` source file. 63
5.4 UML Class Diagram for Grasp Planning Engine package. 64
5.5 Dependency graph for `gp_node.cpp` source file. 65
5.6 Robotiq 2F85 gripper with graspable region between the antipodal grasping plates. 65
5.7 Simple gripper with graspable region between the antipodal grasping plates. 65
5.8 Small frog 3D model used to generate sample outputs. 66
5.9 Example depth map output. 66
5.10 Example segmentation mask output. 66

6.1 Envisat graspability as analysed by voxel superimposition method (#1) with Reward = 1, and Penalty = 50. 71
6.2 Local view of best grasp candidate for method #1 analysis. Grasp quality = 0.833333. 71
6.3 Envisat graspability as analysed by coplanarity of contact points method (#5) with Bin saturation = 50 and Bin steps = 100 72
6.4 Envisat graspability as analysed by voxel superimposition + coplanarity of contact points (#7) with Reward = 1, Penalty = 10, Bin saturation = 50, and Bin steps = 100. 72
6.5 Custom ABB YuMi Gripper used by UC Berkeley for its GQCNN and Dex-Net projects. Graspable region (only area with padding) is approximately $5 \times 3 \times 2$ cm. 74

6.6 Representation of the custom ABB YuMi Gripper used by UC Berkeley for its GQCNN and Dex-Net projects. Graspable region is $5 \times 3 \times 2$ cm. 74
6.7 Overview of the adversarial pyramid object. Its length is 0.5 m, while the diameter ranges from 0.02 m to 0.2 m. 75
6.8 Best grasping candidate of the adversarial pyramid using the custom ABB YuMi gripper model and grasp planning algorithm #1. Right image is isolated view of target voxels (green) interacting with the gripper. Grasp quality score = 0.608333. 75
6.9 Best grasping candidate of the adversarial pyramid using the custom ABB YuMi gripper model and grasp planning algorithm #5. Right image is isolated view of target voxels (green) interacting with the gripper. Grasp quality score = 0.9. 76
6.10 Best grasping candidate of the adversarial pyramid using the custom ABB YuMi gripper model and grasp planning algorithm #7. Right image is isolated view of target voxels (green) interacting with the gripper. Grasp quality score = 0.691667. 76
6.11 Global graspability map of the adversarial pyramid using the custom ABB YuMi gripper model and grasp planning algorithm #1. 77
6.12 Global graspability map of the adversarial pyramid using the custom ABB YuMi gripper model and grasp planning algorithm #5. 77
6.13 Global graspability map of the adversarial pyramid using the custom ABB YuMi gripper model and grasp planning algorithm #7. 78
6.14 Best grasping candidate of the adversarial pyramid using the custom ABB YuMi gripper model ran on the Dex-Net 4.0 GQCNN-4.0-PJ deep learning model from UC Berkeley. Grasp quality score = 0.882. 78
6.15 Illustrated best grasping candidate of the vase object using the custom ABB YuMi gripper model and grasp planning algorithm #7. Grasp quality score = 0.8 79
6.16 Best grasping candidate of the vase object using the custom ABB YuMi gripper model ran on the Dex-Net 4.0 GQCNN-4.0-PJ deep learning model from UC Berkeley. Grasp quality score = 0.905. 79

7.1 Overview of the test environment in GRALS, with the components labelled (computing equipment not in view). 82
7.2 GNC Rendezvous, Approach and Landing Simulator overview. 83
7.3 Validation test payload. 84
7.4 1:10 scaled model of Envisat used in the SIL test campaign. 85
7.5 Short-range spherical trajectory executed during the SIL tests, all units in mm. 85
7.6 Global reconstructed point cloud complied with the K-NN outlier removal method using the 5 and 30 nearest neighbours 86
7.7 Voxel models from single point cloud snapshots of Envisat taken at a distance of 1.29 m, voxel leaf sizes of 0.002 m (left) and 0.005 m (right). 87
7.8 Envisat (1:10 model) reconstruction at a distance of 15.48–13.16 m, voxel leaf size of 0.005 m 88
7.9 Envisat (1:10 model) reconstruction at a distance of 8.39–6.07 m (left), and 6.02–3.70 m (right), voxel leaf size of 0.005 m. 89

7.10 Envisat (1:10 model) reconstruction at a distance of 3.66–1.34 m (left), and 1.29–0.82 m (right), voxel leaf size of 0.005 m. 89
7.11 Global grasp planning of Envisat (1:10 model) reconstructed from a single point cloud snapshot at a distance of 1.29 m (left) and 10 snapshots at a distance of 1.29–0.82 m (right), voxel leaf size of 0.002 m. 91
7.12 Global grasp planning of Envisat (1:10 model) reconstructed from a single point cloud snapshot at a distance of 1.29 m (left) and 10 snapshots at a distance of 1.29–0.82 m (right), voxel leaf size of 0.003 m. 92
7.13 Global grasp planning of Envisat (1:10 model) reconstructed from a single point cloud snapshot at a distance of 1.29m (left) and 10 snapshots at a distance of 1.29–0.82m (right), voxel leaf size of 0.004m 92
7.14 Global grasp planning of Envisat (1:10 model) reconstructed from a single point cloud snapshot at a distance of 1.29 m (left) and 10 snapshots at a distance of 1.29–0.82 m (right), voxel leaf size of 0.005 m. 93
7.15 Global grasp planning and best grasp candidate of Envisat without upscaling and maintaining the standard gripper width of 8.4 cm. Reconstructed from 10 snapshots at a distance of 1.29–0.82 m, voxel leaf size of 0.005 m. 93
7.16 Most graspable region of the unscaled Envisat model analysis (viewpoint 1). 94
7.17 Most graspable region of the unscaled Envisat model analysis (viewpoint 2). 94

A.1 UML State Diagram for the Global Grasp Planner in the case of a global grasp analysis. 109
A.2 Envisat 3D Wavefront model. 110
A.3 Envisat binary tree (voxel) model with a leaf size of 0.0196043 m. . . . 110
A.4 UML Sequence Diagram for 3D to depth map utility. 112

B.1 R61 Mk0 viewpoint 1. 114
B.2 R61 Mk0 viewpoint 2. 114
B.3 R61 Mk1 viewpoint 1. 115
B.4 R61 Mk1 viewpoint 2. 115
B.5 R61 Mk2 viewpoint 1. 116
B.6 R61 Mk2 viewpoint 2. 116
B.7 R61 Mk3 viewpoint 1. 117
B.8 R61 Mk3 viewpoint 2. 117
B.9 R82 Mk0 viewpoint 1. 118
B.10 R82 Mk0 viewpoint 2. 118
B.11 R82 Mk1 viewpoint 1. 119
B.12 R82 Mk1 viewpoint 2. 119
B.13 R83 Mk0 viewpoint 1. 120
B.14 R83 Mk0 viewpoint 2. 120
B.15 R83 Mk1 viewpoint 1. 121
B.16 R83 Mk1 viewpoint 2. 121
B.17 R84 Mk0 viewpoint 1. 122
B.18 R84 Mk0 viewpoint 2. 122

B.19 R84 Mk1 viewpoint 1. 123
B.20 R84 Mk1 viewpoint 2. 123
B.21 R85 Mk0 viewpoint 1. 124
B.22 R85 Mk0 viewpoint 2. 124
B.23 R85 Mk1 viewpoint 1. 125
B.24 R85 Mk1 viewpoint 2. 125
B.25 R86 Mk0 viewpoint 1. 126
B.26 R86 Mk0 viewpoint 2. 126
B.27 R86 Mk1 viewpoint 1. 127
B.28 R86 Mk1 viewpoint 2. 127
B.29 R87 Mk0 viewpoint 1. 128
B.30 R87 Mk0 viewpoint 2. 128
B.31 R87 Mk1 viewpoint 1. 129
B.32 R87 Mk1 viewpoint 2. 129

List of Tables

2.1 Historical comparison of uncooperative autonomous capture of generic space objects missions . . . 10
2.2 Overview of the characteristics of close range 3D imaging sensor technologies. . . . 12
2.3 Typical close proximity rendezvous sensor accuracy requirements . . . 13
2.4 Comparison of storage size required in multiple representations for high-fidelity terrain reproduction, from Gebhardt, Payzer, Salemann, *et al.* [25] 17
2.5 Most influential proposals in the field of grasp planning. Extracted from academic.microsoft.com in August 2020. . . . 22
2.6 Most influential publications using empirical approaches for grasping unknown objects . . . 23
2.7 Collection of linear and non-linear features relevant to spatial histogram grasp quality analysis. . . . 26

3.1 Stakeholders and their roles for the Aut-O-MAGIC project. . . . 35
3.2 Aut-O-MAGIC Key Performance Parameters and Performance Drivers. 41
3.3 Aut-O-MAGIC Acceptance Criteria. . . . 41

4.1 State-of-the-art camera-type 3D sensors specification trade-off . . . 47
4.3 Overview of grasp quality methods proposed for the Aut-O-MAGIC project. 57

6.1 Parameter matrix of the parameters used for the qualitative analysis of Envisat. . . . 70

7.1 Trajectory segments used for surface reconstruction. . . . 87
7.2 Reconstructed voxel models from point cloud data at different distances from the target. . . . 88
7.3 Results of the validation analyses of Envisat at different leaf sizes (resolutions). Envisat was reconstructed from a single point cloud snapshot at 1.29 m distance and a set of point cloud snapshots at 1.29–0.82 m distance. . . . 90

8.1 Technical requirements compliance list. . . . 97

A.1 Command-Line Interface arguments for the Grasp Planning Engine entry point. . . . 108

List of Acronyms

Aut-O-MAGIC	Autonomous On-Orbit Manipulability Analyser for Gripper Interaction Capabilities.
BBX	Bounding Box.
CLI	Command-Line Interface.
COTS	Commercial Off-The-Shelf.
ESA	European Space Agency.
FOV	Field Of View.
FPS	Frames Per Second.
GDB	GNU Debugger.
GNC	Guidance Navigation and Control.
GPS	Global Positioning System.
GRALS	GNC Rendezvous, Approach and Landing Simulator.
GUI	Graphical User Interface.
I/O	Input/Output.
ICP	Iterative Closest Point.
IMU	Inertial Measurement Unit.
IR	Infra-red.
K-NN	k-Nearest Neighbours.
K4A	Azure Kinect DK.
KPP	Key Performance Parameters.
LIDAR	Laser Imaging, Detection, and Ranging.
MEV-1	Mission Extension Vehicle 1.
MEV-2	Mission Extension Vehicle 2.

NFOV	Narrow Field of View.
OOD	Object-Oriented Design.
ORGL	Orbital Robotics and GNC Laboratory.
PCL	PointCloud Library.
PLC	Programmable Logic Controller.
RGB	Red Green Blue additive color model.
RGB-D	RGB-Depth (Time-of-flight).
ROS2	Robot Operating System 2.
RvD	Rendezvous and Docking.
RvD/B	Rendezvous and Docking/Berthing.
RvD/C	Rendezvous and Docking/Capture.
SIL	Software-In-the-Loop.
SLAM	Simultaneous Localisation and Mapping.
UML	Unified Modelling Language.
WFOV	Wide Field of View.

List of Scripts

5.1 Contents of intrinsic camera parameters output file `virtualcam.intr` 66
6.1 Intrinsic camera parameters for the PhotoNeo Phoxi S. camera used to train the CQCNN-4.0-PJ model 73
A.1 3D model voxeliser pipeline (pseudo-Bash commands) 110

I

Introduction and Overview

1 Introduction

1.1. Motivation

In the present day, it has become increasingly common to perform space Rendezvous and Docking/Berthing (RvD/B) manoeuvres using either manual control or through discrete-event controllers. Nevertheless, manual control exposes the operators to danger; while sequenced controllers require either well-defined environments or a cooperative target. Examples of sequenced controllers with well-defined environments are the Space Shuttle's Rendezvous and Docking capabilities, that used the TriDAR 3D sensor, and ESA's CX-OLEV mission extension vehicle concept. Meanwhile, the Soyuz/Progress Kurs docking navigation system relies on docking to cooperative targets. Platform independence from the target spacecraft's architecture can be achieved by using a gripper to grasp the target from a convenient point (i.e. the nozzle) in so-called capture manoeuvres. This allows using a range of non purpose-built surface features as contact locations such that a successful mating of the two objects is not constrained by docking interface compatibility. The increasing interest in satellite robotic operations clearly shows that a system capable of performing an autonomous rendezvous and capture of a novel (never-seen-before) space object can prove advantageous in terms of cost, schedule, and reliability.

Current methods for autonomous RvD/C used by industry rely on preprogrammed instructions by the engineers. These instructions determine the gripping point on the target vehicle, as well as an approach strategy that meets a series of preconceived requirements tailored to a specific target vehicle. Evidently, this requires prior knowledge of the target spacecraft. To achieve unsupervised rendezvous and capture without any prior knowledge of the targets geometric characteristics, it becomes important to develop cognitive systems capable of systematically perceiving relevant features from the target body that can be used as gripping points. This is the main goal of this project.

1.2. Overview

Consider a servicing spacecraft that is to perform maintenance on a malfunctioning satellite. To rendezvous and successfully capture the target satellite as a preparation for the repair work, it requires knowledge of its relative pose and target contact location for capture. While the relative pose can be obtained by multiple means, such as with GPS transceivers on both satellites in the case of a cooperative target satellite, it must be determined without relying on any instruments on the other satellite when dealing with a uncooperative (unresponsive) satellite. Similarly, the contact location for capture on the target can normally be determined a-priori by using production 3D models of the satellite or analytically by the mission control team using images from the servicing spacecraft. These methods, however, require up to date knowledge of the surface geometric characteristics of the target or continuous communication with the mission control team. These limitations justify the development of a solution that enables local grasp planning of satellites for which we do no longer possess accurate surface geometric information. The solution proposed in this thesis is named Autonomous On-Orbit Manipulability Analyser for Gripper Interaction Capabilities (Aut-O-MAGIC).

Although grasp planning could be performed using two-dimensional images, 3D reconstruction of the scene is necessary to determine relative pose between the chaser and the target satellite. Relative pose estimation is crucial to effectively rendezvous and capture an uncooperative satellite, as there is no communication that can otherwise provide the missing relative positioning information. Therefore, in this thesis only 3D imaging techniques are considered.

1.3. Research Objective and Scope

The research context, illustrated in Figure 1.1, covers all the elements that comprise the topic of uncooperative rendezvous and capture. In short, an autonomous, uncooperative, RvC can be envisioned as a system that is capable of sensing the environment (using sensors), and acting upon it (via actuators) with the goal of mating a chaser space vehicle with its target. To act on the environment based on the sensor input, it is necessary to process said input and then fit it into a control strategy that converts it into actuator commands. The processing step consists on building a 3D model of the target using sensor data. This model enables grasp location planning and relative pose estimation, which are used as inputs to the control strategy. This control strategy comes in the form of Guidance Navigation and Control (GNC) functions. These GNC functions are normally designed as discrete-event controllers, which translate the processed input into actuator commands differently depending on the current event. These events are categorised depending on the approach constraints and the phase of the RvC the chaser vehicle is currently in (dependant on the relative distance).

In this context, research questions were devised that are to be answered in reaching the project goal using a systematic approach based on procedural discovery of the research framework of the project and using the SMART principle. The resulting research framework is depicted in Figure 1.2 and defined as follows:

> A study of the various aspects of the research project, from empirical data of previous robotic missions and the consultation of scientific knowledge, yields the list of requirements and the validation procedures, by means of which the novel autonomous grasp planning algorithm will be evaluated. A confrontation of the results of these evaluations conclude with recommendations for developing a capture system with autonomous grasp planning.

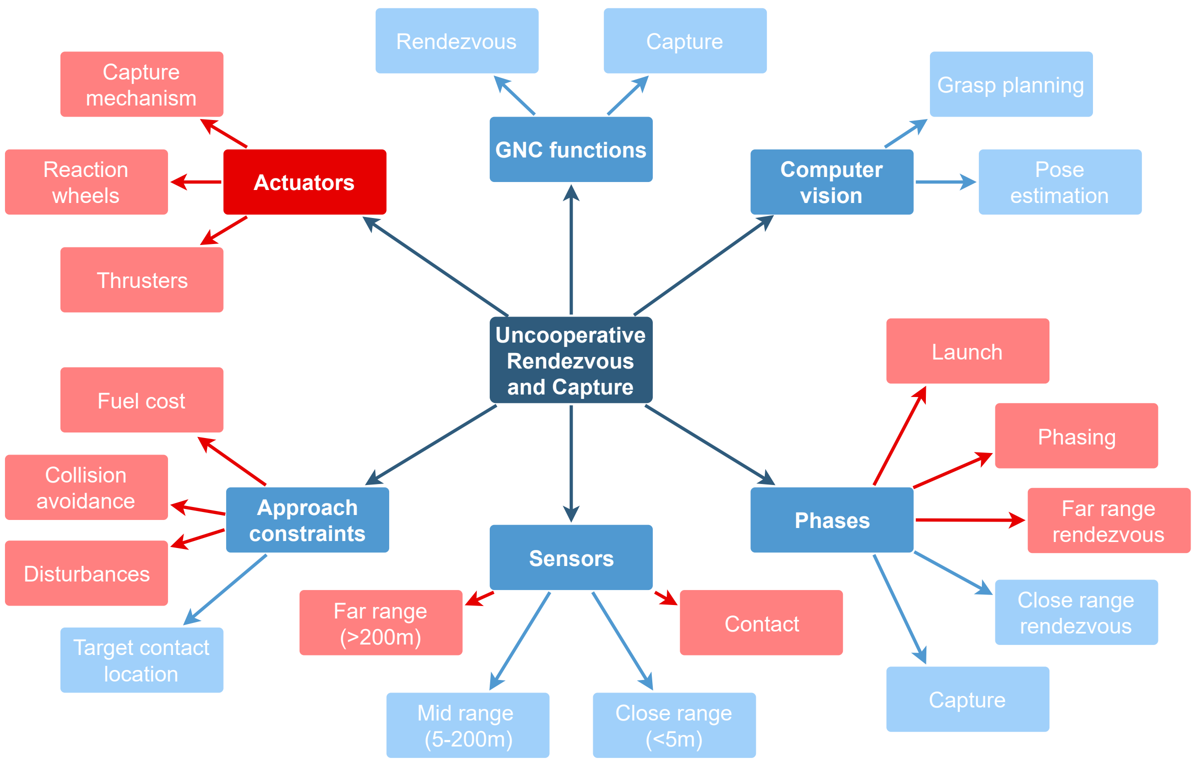


Figure 1.1: Mind map illustration of uncooperative Rendezvous and Capture research context. In blue are the elements treated in this thesis.

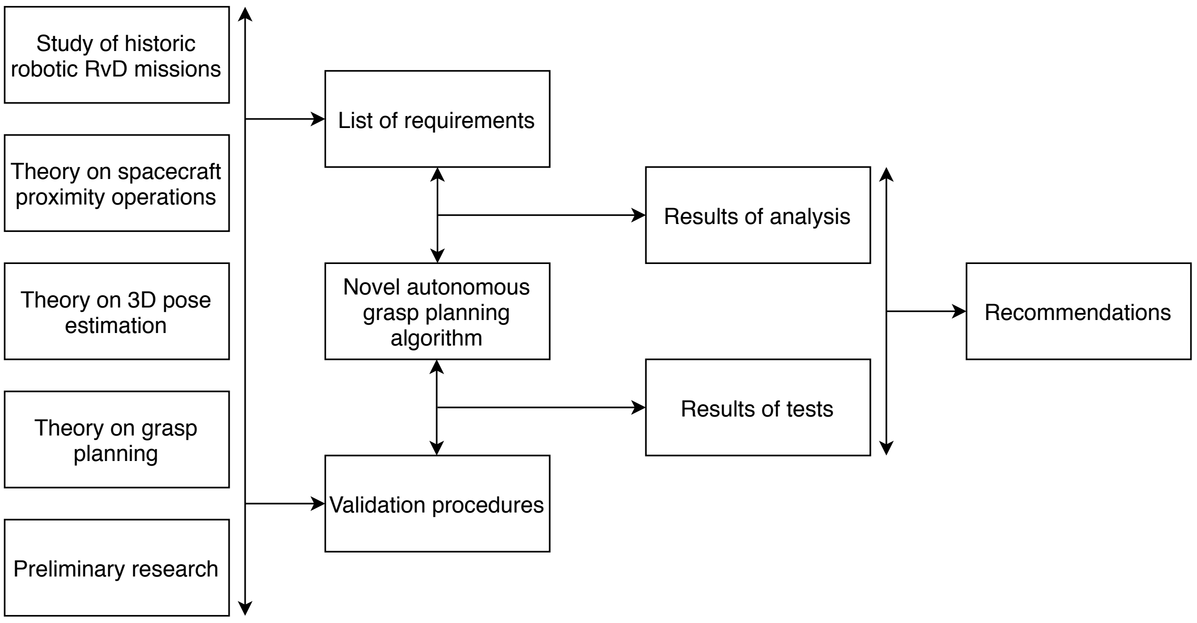

Study of historic robotic RvD missions
Theory on spacecraft proximity operations
Theory on 3D pose estimation
Theory on grasp planning
Preliminary research
List of requirements
Novel autonomous grasp planning algorithm
Validation procedures
Results of analysis
Results of tests
Recommendations

Figure 1.2: Research framework 'Autonomous Grasp Planning for Robotic Capture of Space Objects'.

From the identified knowledge gaps in literature and the research framework, the central question to be solved with the completion of this thesis project is:

> **How can the performance of current analytic autonomous grasp planning methods for terrestrial applications be improved to enable robust on-orbit grasp planning of uncooperative novel satellites?**

This is then broken down into parts and sub-parts, which correlate to the divisions depicted in the research framework:

1. What requirements are relevant for evaluating the performance of an autonomous space grasp planning system?
    (a) Which resolution is required to achieve good results?
    (b) What noise levels does the system need to reliably operate under?
    (c) What constitutes a successful grasp?
2. Which validation procedures most truthfully recreate the operating conditions of the system in space?
3. How can geometric matching using 3D maps provide a performance advantage compared to image-based analyses?
    (a) What are the limitations of image-based grasp planning?
    (b) How does the computational efficiency of both methods compare?
    (c) Do less-occluded 3D maps provide a robust knowledge advantage over single-view depth images?

2
Grasp Planning Pipeline Overview

To justify the design effort of this thesis, this chapter treats the background and current state-of-the-art in space proximity operations and grasp planning. On the closing section, a conclusion on the improvement gaps available for on-orbit grasping operations is presented.

2.1. Space Proximity Operations

Space proximity operations encompass sensing, control, and actuation technologies to enable the overall activity. Because of the contents and scope of this thesis, we focus on the final phase of the rendezvous and its associated short-range sensors. Space proximity operations have historically not attempted to perform all the aspects involved in uncooperative autonomous capture of generic space objects, which warrants a discussion on the opportunities and challenges associated with performing all those aspects.

2.1.1. Rendezvous and Docking

Space rendezvous and docking operations are a key technology necessary for most missions that require interaction of multiple space objects. The first operation of this type was performed on the 16th of March 1966 during the Gemini VIII mission by successfully docking an active vehicle, commonly called the *chaser*, with a stabilised passive *target* (Figure 2.1). Since then, this technique has been used by numerous programs, including other Gemini missions (1966), Apollo (1968–1972), Salyut (1971–1986), Skylab (1973–1974), Apollo/Soyuz (1975), Mir (1986–2000), ISS (2000–), and Tiangong (2012–2019) programs [1]–[4]. In the topic of uncooperative rendezvous and capture, missions that either achieved an uncooperative docking, or performed capture without a docking interface are considered to be of relevance. An uncooperative docking is defined as a docking without visual aids, communication, or active attitude control of the target spacecraft.

Unlike the United States, which used manual piloted docking throughout the Apollo, Skylab, and Space Shuttle programs, the Soviet Union employed automated docking systems from the beginning of its docking attempts. An exception to this practice was the mission Soyuz-T 13, where the crew manually docked with the inert Salyut 7

Figure 2.1: View from the nose of Gemini VIII as it approaches the docking collar of the Agena target vehicle as Neil Armstrong and David Scott complete the world's first Rendezvous and Docking between two spacecraft in orbit. Credits: NASA/David Scott.

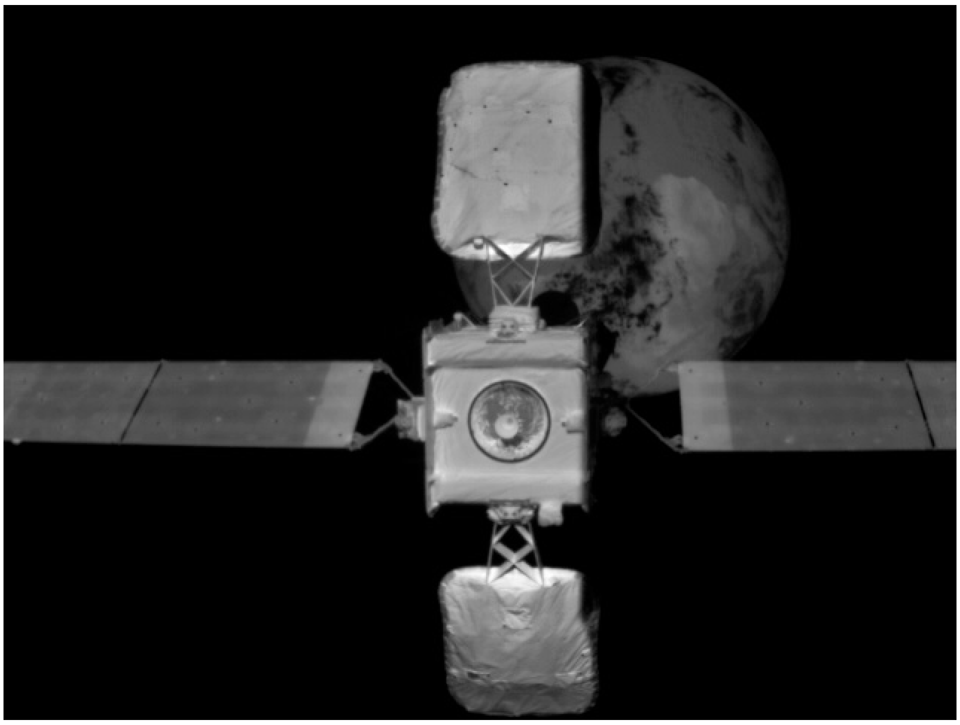


Figure 2.2: MEV-2 IR WFOV camera view at 15m away from Intelsat 10-02 in 2021 [5].

space station with the goal of repairing it [6]. This is the first time an uncooperative RvD was ever performed. Since then, only one other mission ever performed an uncooperative docking with another spacecraft. The Orbital Express mission was executed in 2007 by the United States and DARPA with the goal of demonstrating the technology necessary for on-orbit servicing. This included autonomous rendezvous and capture, subsystem replacement, refuelling, and an autonomous fly-around visual inspection [7]. The only pitfall of this mission compared to an complete implementation of on-orbit servicing is that the target satellite used was specifically designed for this mission and therefore was optimised for robotic capture.

Meanwhile, there is a higher number of missions that conceptualised or executed a robotic capture of a cooperative target. The first ever series of cooperative robotic captures were executed by Space Shuttle crews, starting with the repair of the SolarMax satellite in 1984 and, subsequently, the Hubble Servicing missions, performed between 1993 and 2009 [2]. Since then, only the Mission Extension Vehicle 1 (MEV-1) and MEV-2 from Northrop Grumman have carried out a cooperative robotic capture. In February 2020 MEV-1 successfully docked with a telecommunications satellite taking over its attitude control and orbit maintenance functions [8]. In April 2021, MEV-2 also accomplished docking (Figure 2.2). In contrast with the 2007 Orbital Express mission, this was the first time a satellite robotic capture was performed with a satellite that was not designed with docking in mind. Further conceptual robotic capture vehicles that never made it to orbit or are still in the development phase include the ESA co-funded ConeXpress orbital life extension vehicle and its subsequent adaptation as SMART-OLEV, and the Swiss ClearSpace One debris removal vehicle (to be launched by 2025). In Table 2.1, these missions are tabulated with an overview of their similarity to this thesis goal and their relevant system architecture.

In practical terms, a rendezvous and capture mission can be divided into a number of phases. These are: launch, phasing, far range rendezvous, close range rendezvous, and capture [2]. Up to the completion of far range rendezvous, the goal effectively lies on transferring the chaser vehicle to the vicinity of the target spacecraft using absolute navigation and far range relative navigation sensors. The transfer trajectory must also take into consideration the approach strategy chosen for close range rendezvous. The two most common approaches are to begin proximity operations from a stable point on V-bar (along the target velocity vector) or to perform a double co-elliptic approach. These approaches have different advantages: the first gives the opportunity to pause the approach without fuel consumption and reassess the situation before close range rendezvous, while the double co-elliptic approach transfers the chaser to a Natural Motion Circumnavigation relative orbit around the target that permits inspecting the satellite from a 360° perspective [15].

Once the chaser reaches a relative distance of 100–200m proximity operations can begin. In the proposed scenario of uncooperative rendezvous and capture, this results in using the 3D imaging sensors to perform accurate pose estimation as well as to develop the model of the target vehicle required for grasp planning.

When the contact location has been decided and the rendezvous conditions are

[1]It is important to note that although the ClearSpace One system does capture the target object, it does so in a rudimentary way that would not allow for servicing or interaction with said vehicle beyond towing

Table 2.1: Comparison of relevant missions in the research context of uncooperative autonomous capture of generic space objects. The generic target qualifier conveys that the approaching spacecraft can perform capture independently of the surface characteristics of the target vehicle (i.e. the gripping interface is not tailored to the target).

Vehicle	Launch date	Uncooperative	Automated	Robotic Capture	Generic target	System architecture	Ref.
Space Shuttle	1984–2009			X	X	6-DOF robotic arm; elbow, wrist and side-view monocular cameras	[9]
Soyuz-T 13	1985	X				Manual control levers, axial video camera, and docking periscope	[6]
Orbital Express	2007	X	X	X		6-DOF robotic arm, force/torque sensor, monocular camera at end effector, and grapple fixture on the target	[10]
ConeXpress	N/A (2007 study)		X	X		Purpose-designed gripping system, close+mid+far range monocular camera set, and ground-based image processing	[11]
SMART-OLEV	N/A (2008 study)		X	X		Purpose-designed gripping system, near field stereo camera, illumination system, laser and inductive capture sensors, and ground-based image processing	[12]
MEV-1, MEV-2	2019, 2020		X	X		Purpose-designed gripping system with WFOV IR and NFOV visual optical sensors	[5], [8]
ClearSpace One	≈2025	X	X	X[1]	X	Conical collapsible net and unspecified optical RvC sensor	[13], [14]

favourable (Sun illumination vector, relative pose) close range rendezvous can continue. Through a closed-loop GNC algorithm, the spacecraft approaches the target until capture. Depending on the approach strategy, this can be performed through a straight line motion or by gradually decaying the circumnavigating relative orbit. Both strategies are possible because the commonly applicable approach corridors (mandatory straight line final approaches for safety) are not required when docking a robotic spacecraft with an unmanned uncooperative target. However, as the field of view of the rendezvous sensor is also limited (usually ± 15 deg), it puts a comparable constraint on the trajectory design [2]. Final approach and capture is typically performed at speeds of 30-60mm/s such that potential plume impingement and momentum exchange at contact is minimised [16].

2.1.2. 3D Imaging for Relative Navigation

Proximity relative navigation in space has been historically performed with monocular and stereo cameras in the IR and visual spectrum (Table 2.1). This is reasonable, because optical sensors are advantageous compared to laser rangers or radars due to their higher accuracy at close ranges. In Figure 2.3, the suitability of the different rendezvous sensors available is clearly illustrated and shows how only optical sensors provide sub-centimetre accuracy at close range.

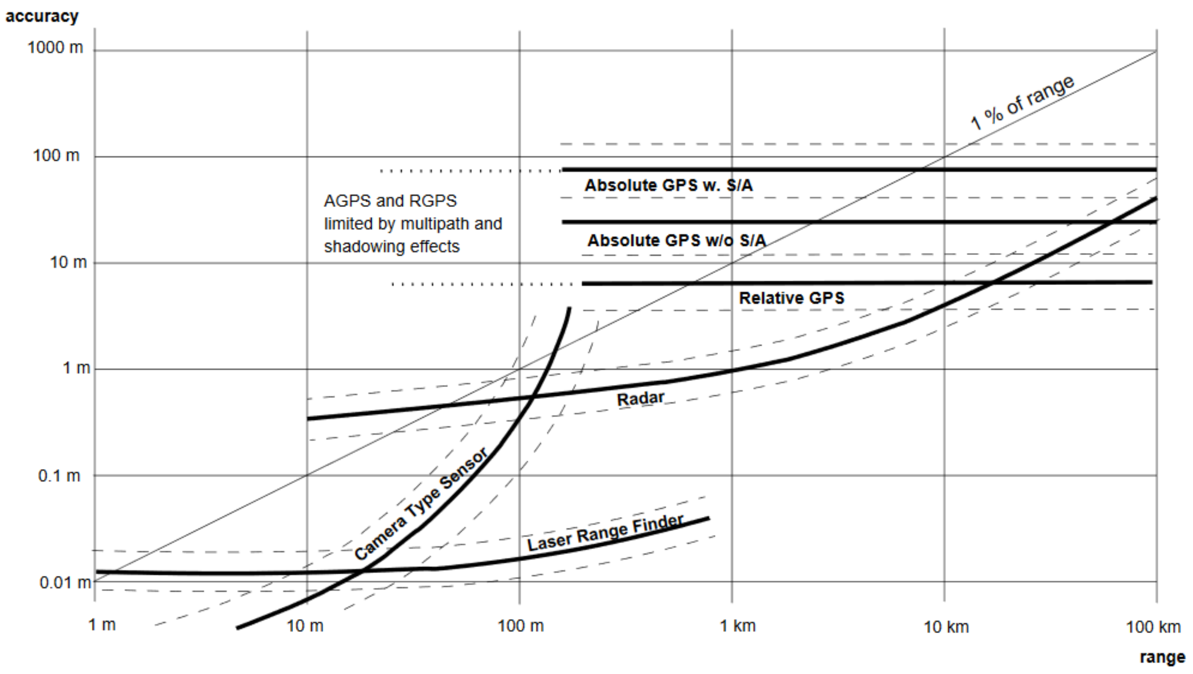


Figure 2.3: Operational range and measurement accuracies of common rendezvous sensors, from Fehse [2].

Camera-type sensors can be further divided into three solutions: monocular cameras, stereoscopic cameras, and time-of-flight cameras. Monocular cameras are inherently two dimensional sensors, but the 3D shapes in the scene can be recovered using multiple images via photometric stereo recovery or through a single image via silhouette techniques such as shading analysis. An overview of the fitness for purpose of each camera-type sensor is provided in Table 2.2. The criteria for comparison are:

- The system **type**: active systems emit energy in order to scan objects and are therefore more sensitive to high background noise levels;

- The state **observability**: whether the sensor allows for a direct determination of the state of a target;

- The point cloud **density**: dense methods versus point-based “sparse” density;

- The **accuracy**: precision on the determination of the state of the target;

- The observable **range**: distance from which depth can be determined.

Table 2.2: Overview of the characteristics of close range 3D imaging sensor technologies.

Sensor technology	Type	State observability	Point-cloud density	Accuracy	Observable distance	Ref.
Monocular vision	Passive	Partial	Dense	Low	Medium	[17]
Stereoscopic vision	Passive	Full	Dense	Medium	Medium	[18]
Time of flight camera (RGB-D)	Active	Full	Dense	Medium	Short	[19], [20]
LiDAR	Active	Full	Sparse	High	Long	[21]

In Table 2.3, the typical pose determination requirements for a RvD mission are tabulated. These are obtained from the $(1 - \sigma)$ requirements of the Orbital Express Advanced Video Guidance Sensor [22] and largely match the generic $(3 - \sigma)$ RvD accuracy guidelines presented by Mitchell [15]. Other sensors in automated RvD vehicles, such as the Videometer in the European ATV, are shown to easily meet and surpass these requirements [23]. It is worth noting, however, that these requirements stem from missions that had knowledge of the target’s geometric characteristics and had visual aids on the target to assist with the relative pose estimation. In the case of an uncooperative rendezvous of a target without pre-existing geometric information these requirements may be more difficult to achieve.

Table 2.3: Typical close proximity rendezvous sensor accuracy requirements. A single sensor is not expected to cover the entire operational envelope. Obtained from the requirements of the Orbital Express Advanced Video Guidance Sensor [22] and validated against the guidelines proposed by Mitchell [15].

Range [m]	Distance Accuracy [m]	Roll [deg]	Pitch, Yaw [deg]
$[1, 3)$[2]	± 0.012	± 0.13	± 0.20
$[3, 5)$	± 0.035	± 0.25	± 0.33
$[5, 10)$	± 0.150	± 0.45	± 0.70
$[10, 30)$	± 0.150	± 0.45	± 0.70
$[30, 50)$	± 0.400	± 0.45	± 1.20
$[50, 100)$	± 1.666	± 0.50	± 2.40
$[100, 300)$	± 15.000	± 1.40	± 7.00

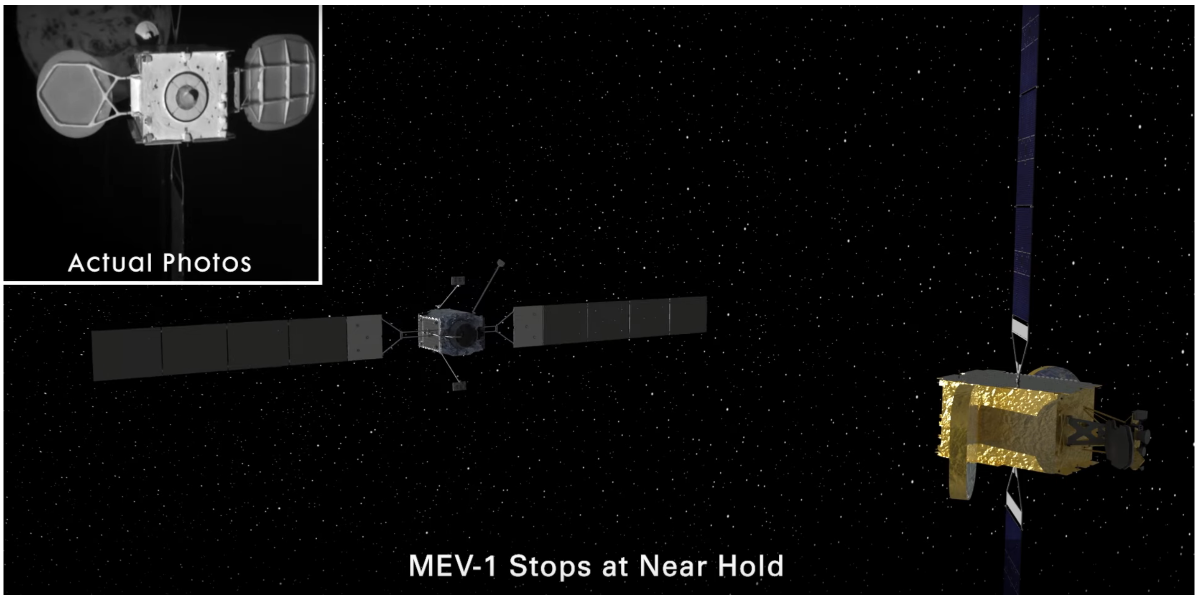


Figure 2.4: Simulation of relative pose between MEV-1 and target Intelsat 901 compared with image taken by IR WFOV proximity camera [8].

2.1.3. Opportunities and Challenges

Grasp planning for space applications has a set of distinct opportunities and challenges compared to grasp planning applied to terrestrial applications. In space, we are able to obtain dense geometric surface information of the grasping target. This is distinctively possible due to the free-floating dynamics of space which gives the chaser the ability to circle around the target and obtain a complete 3D model. Nevertheless, grasping also requires accurate surface mapping, matching a resolution of at

[2][*first*, *last*) Is used to indicate whether the range includes or excludes the number, respectively. I.e. [1,5) = 1, 2, 3, 4

least 10cm to enable grasping of primitive shapes (given a standard grasping width of 20cm), and a higher resolution for thin graspable structures. Satisfactory resolution is more challenging to obtain in the space environment than in terrestrial applications due to the changing environmental conditions, such as lighting conditions, availability of suitable sensors for space applications, and very high reflectivity of some surfaces in satellites—which decrease performance of some sensors such as time-of-flight cameras. Lastly, because satellites are highly valuable, careful selection of grasping point is warranted to avoid damaging sensitive surfaces or instruments. This imposes a challenge that is material recognition and/or sub-system segmentation to filter out grasp candidates in fragile areas.

2.2. Data Structure

When obtaining 3D measurements from the on-board sensors, it is often advantageous to compile them into a 3D map. This allows for analyses that require more than just what the sensors see at that immediate instance. Some examples are volumetric analyses, global grasp planning, and Simultaneous Localisation and Mapping (SLAM) algorithms.

There are two common ways of representing data in 3D space: 3D points in a point cloud, and voxels on a regular grid in 3D space. Sometimes polygons can also be used to describe 3D shapes, but they are only worthwhile for simple homogenous structures and they have to be generated through computationally expensive surface reconstruction algorithms. This work focuses on 3D mapping for space applications, where shapes are large and complex. Therefore, polygon representations can be safely disregarded.

2.2.1. Point Cloud

Point clouds are discrete 3D measurement points that can be collected from a variety of sensors, such as laser range scanners, stereo cameras, or depth cameras. Depending on the sensor used these points may possess more or less attributes. In the case of laser range scanners, their only properties are X, Y, and Z position. With camera-based sensors, it is possible for them to also contain colour information. Figure 2.8 depicts a point cloud generated in the Orbital Robotics and GNC Laboratory at ESA/ESTEC using the Azure Kinect DK (K4A) depth camera (see Figure 2.5 to Figure 2.7). It can be observed how there are occlusions caused by the viewpoint used and low reflectivity of some materials. These shortcomings can be remedied by capturing multiple snapshots and stitching them together, which is be described in subsection 4.5.5. Point clouds are the data that can be directly obtained from 3D sensors. Other 3D representations require processing to convert them from the point cloud representation. Point clouds are unorganised, discrete, non-equidistant, 3D representations. However, they store the *raw* observation from the sensors. As such, they can never be disregarded and for some applications they may be the optimal data representation.

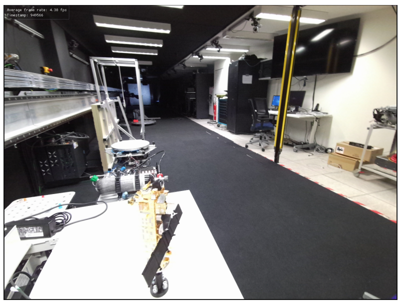


Figure 2.5: K4A colour image.

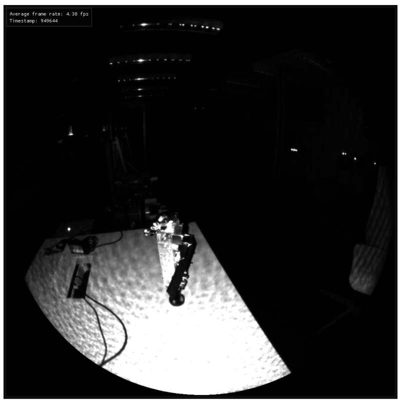


Figure 2.6: K4A IR image.

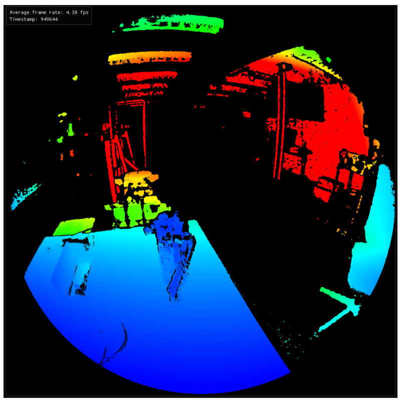


Figure 2.7: Computed 2D depth image form colour and IR image.

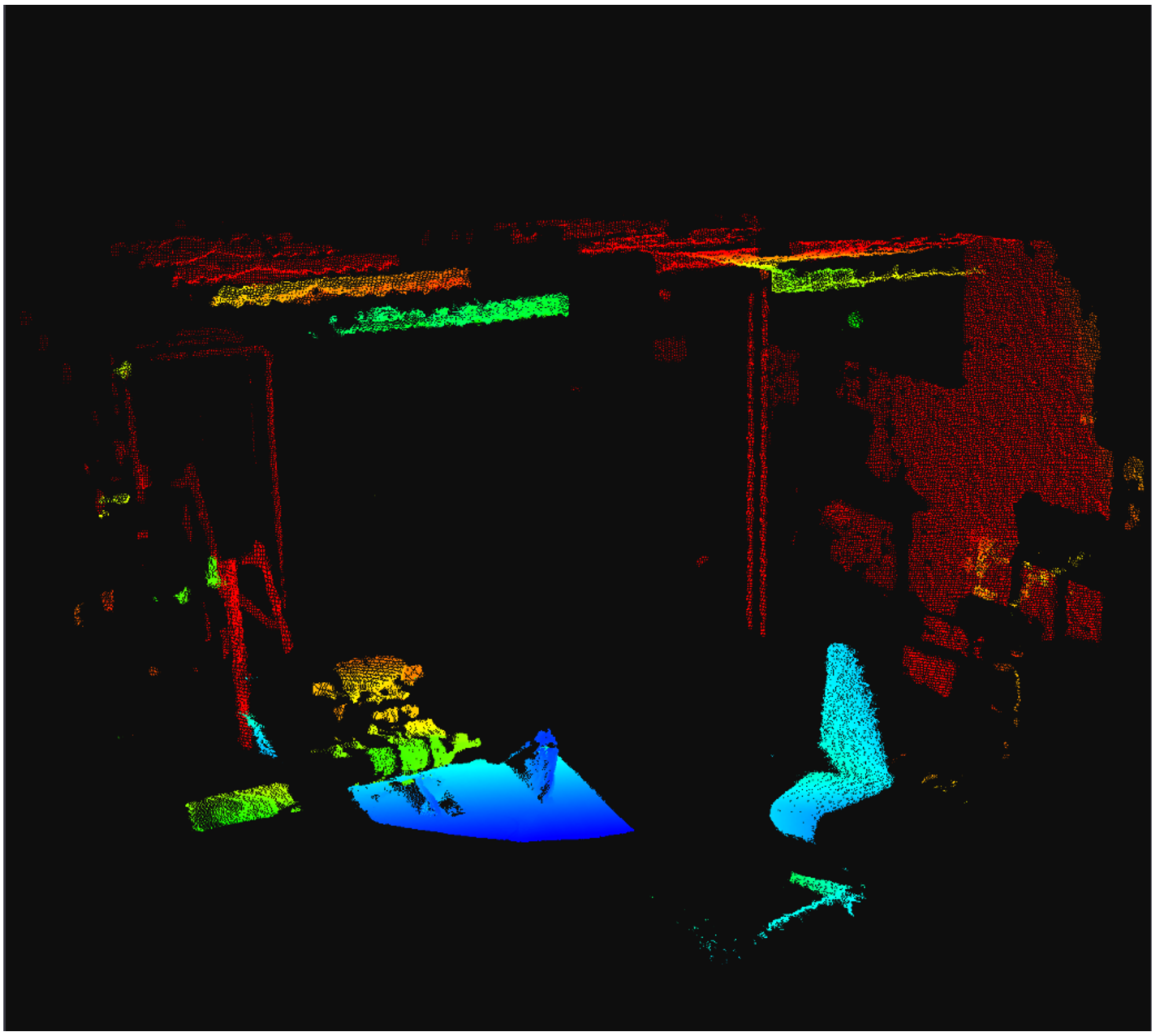


Figure 2.8: 3D depth point cloud projected from depth image depicted in Figure 2.7.

2.2.2. Octree

Voxels can be envisioned as volumetric (3D) pixels. They occupy a cubical space of a certain size, and can encode any information within them, physical or otherwise (colour, opacity, density, location, etc). They are usually generated from point clouds, and their utility is threefold:

1. Provide ability to adapt 3D map resolution to performance/cost requirements through down-sampling.
2. Rasterise[3] 3D map representation to simplify the dataset and obtain equidistant sampling (point clouds may have uneven point set density) [24].
3. Organise the data by indexing the un-organised point set into the generated voxels in a tree structure [24].

Figure 2.9 illustrates how the resolution affects the details observable in a curved object, such as a torus. As it can be observed, the characteristics of the object are greatly disturbed by low resolution representations. For this reason, different voxel models exist [24]:

- Gridded voxel model: depicted in Figure 2.10, this model slices the workspace into voxels of equal sizes, regardless of their internal properties (or lack of).
- Sparse voxel model: depicted in Figure 2.11, this model only generates voxels of a predefined size for regions that contain information.
- Octree voxel model: depicted in Figure 2.12, this model combines neighbouring voxels of identical properties into a single, larger, voxel. It is generated procedurally top-down, with each large voxel dividing into eight children voxels if certain conditions are met (such as number of data points within the voxel volume).

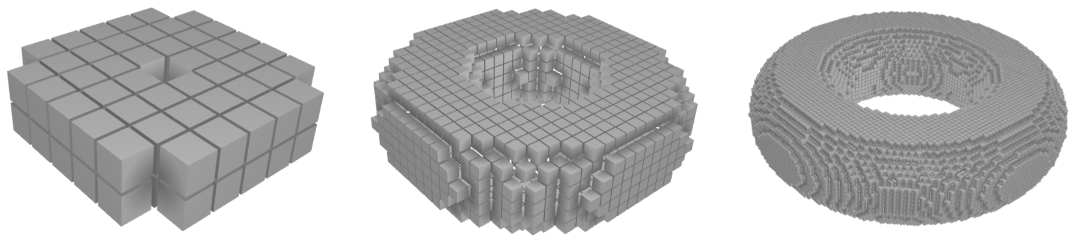


Figure 2.9: A torus represented by 100, 1k, and 10k voxels.

These voxel models have different storage size constraints, described in Table 2.4, which are part of the trade-off to perform in the selection of the optimal voxel model for a given application.

[3]Rasterisation in this context is the process of computing the mapping from scene geometry with irregular density distributions to regularly-spaced measurements

Table 2.4: Comparison of storage size required in multiple representations for high-fidelity terrain reproduction, from Gebhardt, Payzer, Salemann, *et al.* [25]. UPC Points are the packaged point cloud files that needs to be imported into the algorithm.

Terrain Model	Storage size [MB]
UPC Points	184
ASCII Point cloud	267
Polygonal/TIN	126
Gridded Voxel	1,600
Sparse Voxel	3.0
Octree Voxel	6.8

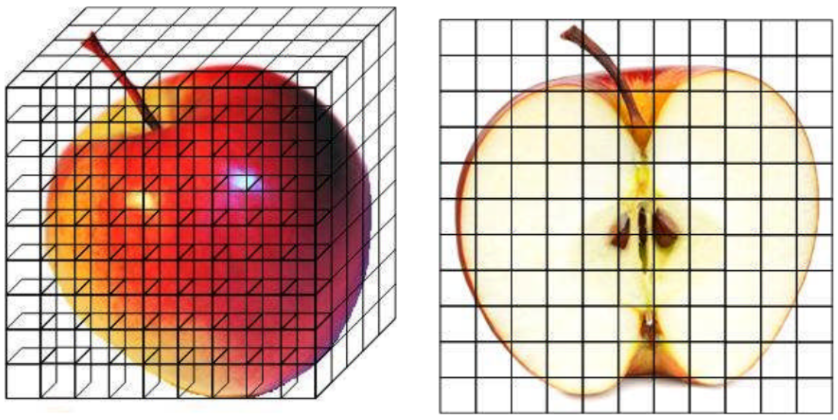


Figure 2.10: Gridded voxel representation, from Gebhardt, Payzer, Salemann, *et al.* [25].

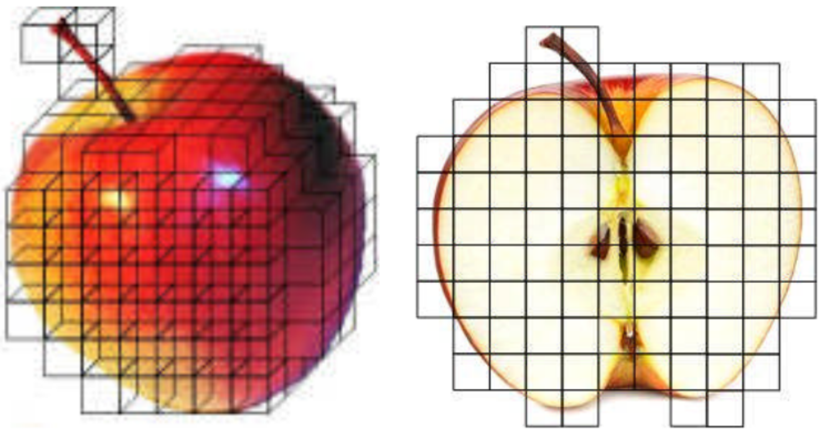


Figure 2.11: Sparse voxel representation, from Gebhardt, Payzer, Salemann, *et al.* [25].

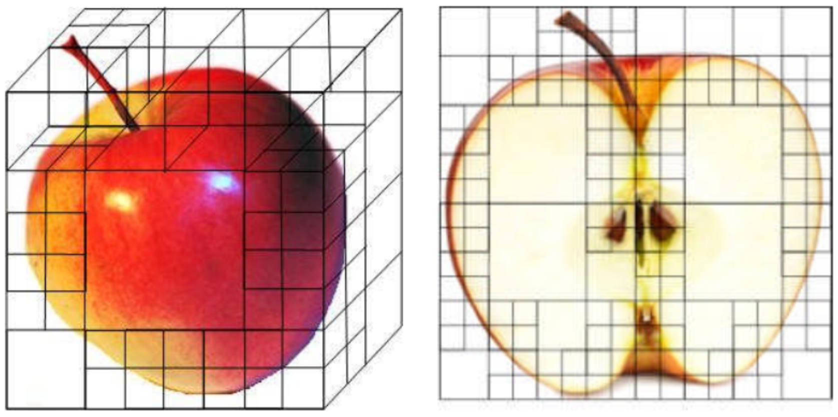


Figure 2.12: Octree voxel representation, from Gebhardt, Payzer, Salemann, *et al.* [25].

2.3. Grasp Planning

Grasp planning is a fundamental problem in autonomous systems that intend to manipulate external objects in a dexterous way. It is considered to be a difficult problem due to the large amount of possible hand configurations as well as the limited and likely imperfect knowledge of target object properties such as shape, pose, material, and mass. The mind map depicted in Figure 2.13 presents all the aspects that play a role in grasp planning. These are:

- Grasp quality metrics: metrics that quantify the probability of success of a certain candidate grasp.
- Grasp quality attributes: distinction between quality metrics that are attributed to the object as a whole (i.e. center of mass) and local ones that only depend on a part of the object (i.e. shape of the surface).
- Grasp candidate generation: way used to instantiate the grasp candidates. It can influence the speed and quality of the grasp planning.
- Preprocessing: techniques to increase the robustness of the grasp planner.
- Object features: information used to determine the quality of a certain grasp candidate. Multiple sources of information can also be combined to yield a multimodal analysis.
- Prior object properties knowledge: the availability of an *a priori* model of the target object can greatly influence the methodology used in grasp quality analyses.

Of the aspects presented above, the grasp quality metrics are the most critical. The other aspects can largely be seen as characteristics of these grasp quality metrics or requirements for the system. These grasp quality metrics can be classified depending on their design approach. Quality metrics determined through empirical (experience-based) methods require large amounts of training data and significant training time, while analytic (rule-based) methods are dependent on the quality of the hand-crafted success metrics and can become computationally expensive due to iteration over numerous grasp candidates. Due to the vast problem space present in grasp planning, this thesis work does not try to treat all the aspects mentioned, but focus on the one with the largest improvement gap for space robotics: grasp quality metrics. The aspects treating grasp candidate generation and preprocessing are also touched upon to a lesser degree as they will contain design decisions for the grasp planning algorithm to be developed. The remaining aspects are merely the selected approaches to solve the problem, as described in previous sections, and do not impose restrictions on the design itself.

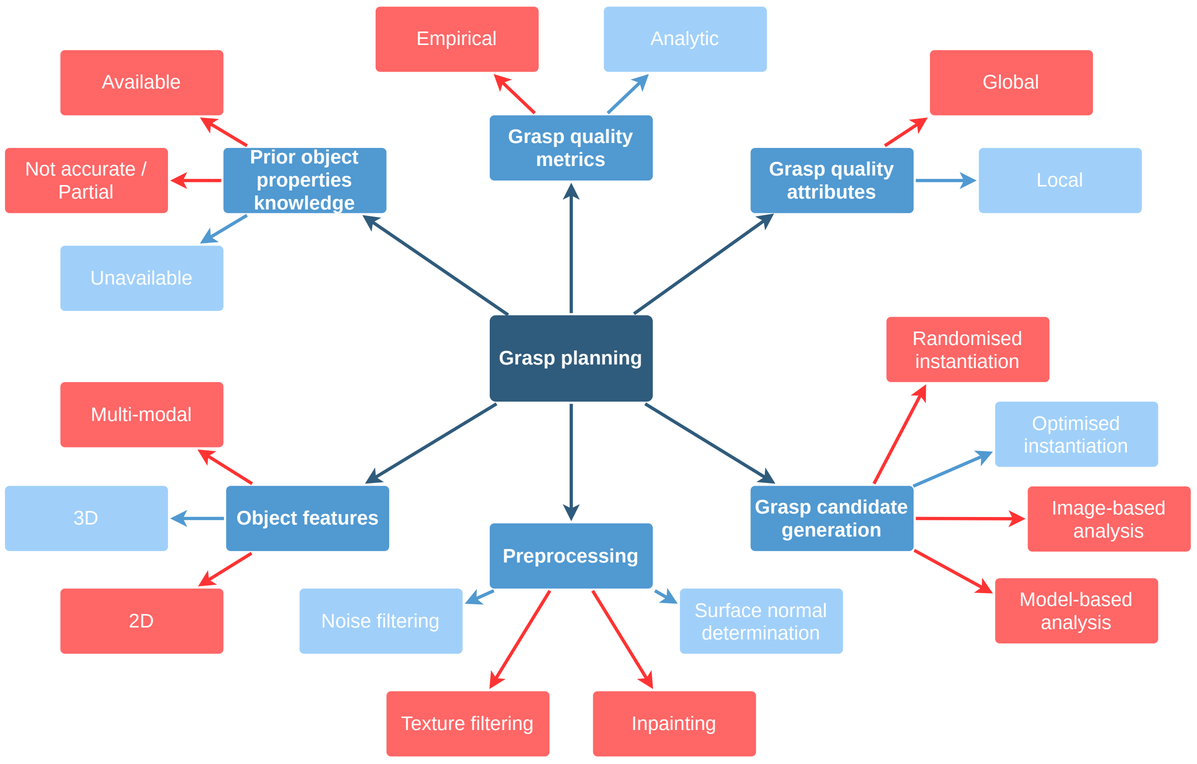


Figure 2.13: Mind map of the aspects that play a role in the problem of grasp planning, with the aspects within the scope of this thesis work in blue background.

2.3.1. Preprocessing

Noise filtering is performed to minimise the number of outliers from sensor data incorporated into the reconstructed 3D model. Outliers can occur due to multiple reasons, including dust particles in the observation area, reflections, and sensor inaccuracies.

A common approach to detect and filter outliers is the k-Nearest Neighbours (K-NN) algorithm, which computes the distances between a certain point and the k nearest neighbouring points. This set of distances is then used to compute the mean μ and the standard deviation σ of the distances. Outliers are then defined as neighbors with a distance from the center point greater than Equation 2.1.

$$\mu \pm \alpha \cdot \sigma \tag{2.1}$$

Where α is a factor that depends on the size of the analysed neighbourhood.

In Figure 2.14, an example K-NN analysis of a 2-dimensional point set is depicted. It can be observed how out of the 5 closest neighbors, one point is classified as an outlier due to its detachment from the other points in the cluster.

Rusu, Marton, Blodow, *et al.* suggest that setting $\alpha = 1, k = 30$ yields good results with approximately 1% of the points classified as outliers [26].

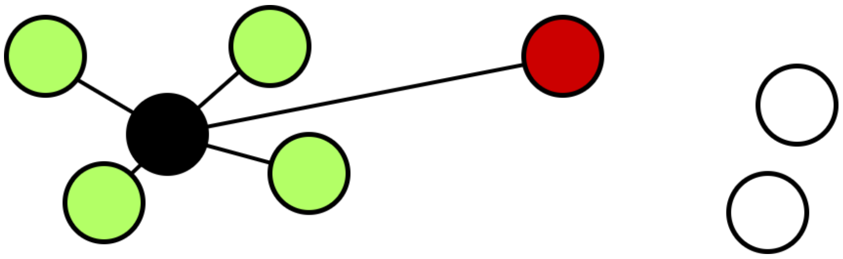


Figure 2.14: Example 2-D k-Nearest Neighbours analysis with $k = 5$, accepted neighbours in light green, rejected outliers in dark red, and nodes not within the 5 closest neighbours in white.

Determination of surface normals from a point cloud can be done as part of the surface reconstruction task or trivially obtained from a fully reconstructed surface. When determined as part of a surface reconstruction task, there are two common robust approaches to solve the problem of surface normal estimation in $\mathbb{R}^3$:

1. Fit a local plane (least squares) using the *k*-nearest neighbors [27]. The parameter *k* can be determined experimentally, although some approaches are also proposed by Hoppe, DeRose, Duchamp, *et al.* on how to determine it automatically. A sensible value, which is the default in Matlab's Computer Vision Toolbox function[4], is $k = 6$.

2. Fit a local plane (least squares) with all points inside a sphere of radius *r*, centred on the point of interest [28]. Mitra and Nguyen describe how the estimation error cannot be arbitrarily minimised if the point set presents noise and curvature. To minimise the error bound, the radius of the sphere should be defined as described by Equation 2.2.

$$r = \left(\frac{1}{\kappa}\left(c_1\frac{\sigma_n}{\sqrt{\epsilon\rho}} + c_2\sigma_n^2\right)\right)^{\frac{1}{3}} \tag{2.2}$$

Where κ is the surface curvature on the point of interest (reciprocal of the radius, i.e. $\kappa = \frac{1}{R}$), σ_n is the noise, ϵ is some small positive number complimentary of the probability that our prediction is within bounds (i.e. probability is at least $1-\epsilon$), ρ is the density of the point cloud, and $\{ c_1, c_2 \}$ are some small constants depending only on the distribution of the point cloud. Below are the conclusions that can be drawn from the above equation for scenarios with high/low noise, curvature and sampling density:

- Low noise or high curvature: $r \to 0$.
- High noise or low curvature: $r \to \infty$.
- High point cloud density: $r \approx c_2\left(\frac{\sigma_n^2}{\kappa}\right)^{\frac{1}{3}}$.

[4] `https://www.mathworks.com/help/vision/ref/pcnormals.html`

- Low point cloud density: $r \approx c_1\left(\frac{\sigma_n}{\sqrt{\epsilon\rho}}\right)^{\frac{1}{3}}$.

Lastly, as two proposed approaches for determining the surface normals only differ in the selection of neighbouring points, they can be both formally defined as follows [28]:

Given a set of *k* points $p_i, 1 \leq i \leq k$, let:

$$\mathbf{M} = \frac{1}{k}\sum_{i=1}^{k} p_i p_i^T - \bar{p}\bar{p}^T \tag{2.3}$$

Where $\bar{p} = \frac{1}{k}\sum_{i=1}^{k} p_i$. Then, the surface normal to the least squares plane formed by this set of *k* points is the eigenvector corresponding to the minimum eigenvalue of **M**.

When determining the surface normals from a voxelised representation, a choice must be made on the breadth of neighbouring voxels to consider. More neighbours will increase the smoothness of the normal field, which inherently reduces noise. However, if the resolution of the voxel 3D map is not significantly higher than the resolution required, this could erroneously determine the normal vector of thin surfaces. In this work, the resolution available is not abundant, so two methods that only consider the immediate neighbours are considered:

- Marching Cubes: generate isosurfaces for every neighbouring voxel relative to the queried point and merge adjacent ones, generating a isosurface normals set up to 15 (unique) cases long, depicted in Figure 2.15 [29];

- Neighbourhood center of volume: given a voxel is on the surface of the model (< 26 neighbours), find the pointing vector connecting the center of the queried voxel to the center of volume of the occupied voxel neighbourhood.

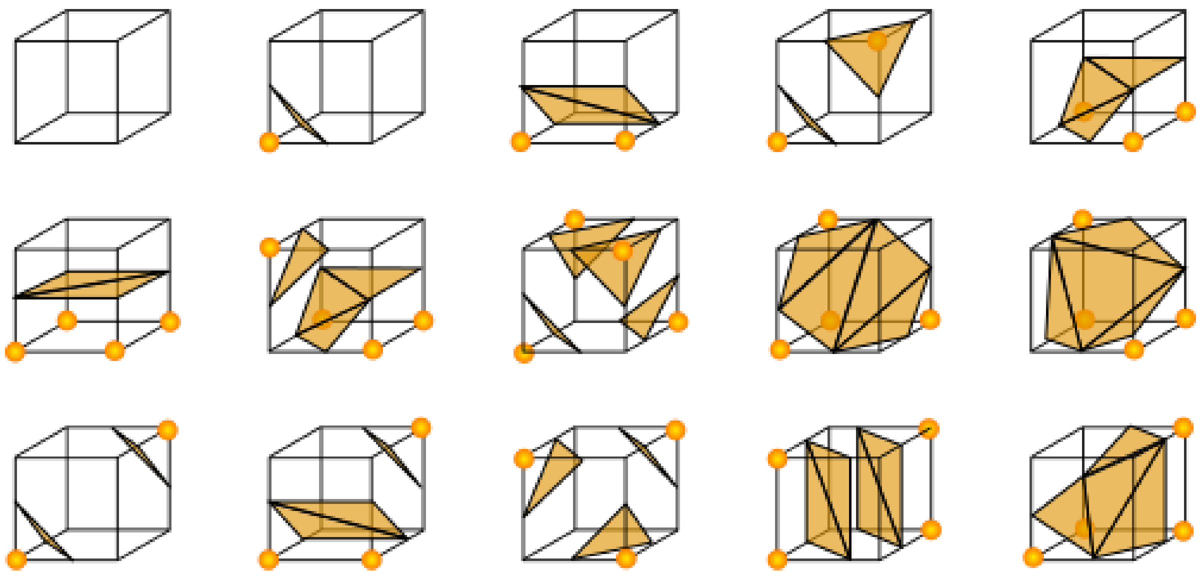


Figure 2.15: The 15 unique isosurface cases in the Marching Cubes algorithm.

2.3.2. Grasp Quality Metrics

The current state-of-the-art in grasp planning is tabulated in Table 2.5. It can be observed how popular methodologies span almost all existing methodologies (relating back to the grasp planning mind map depicted in Figure 2.13). This implies that there is no globally optimal grasp planning method and the performance of one over another one is highly dependant on the task at hand. For the purpose of grasp planning of a novel (generic) object, Bohg, Morales, Asfour, *et al.* in 2014 performed an exhaustive survey of the state of the art of empirical methods. The results of the survey are tabulated in Table 2.6 and show an absolute preference for analytic grasp quality metrics over other empirical solutions[5]. This further strengthens the hypothesis that optimal grasp planning method depends on the task at hand.

Table 2.5: Most influential proposals in the field of grasp planning. Extracted from academic.microsoft.com in August 2020.

Author	Year	Citation count	Methodology
Nguyen [31]	1988	1253	Analytic → Force closure
Ferrari and Canny [32]	1992	957	Analytic → Force closure
Lenz, Lee, and Saxena [33]	2015	835	Empirical → Human labels
Miller, Knoop, Christensen, *et al.* [34]	2003	756	Analytic → Spatial histogram features + Hill-like 3D surfaces
Redmon and Angelova [35]	2015	353	Empirical → Human labels
Jiang, Moseson, and Saxena [36]	2011	277	Analytic → Shape segmentation with graspability metrics
Mahler, Liang, Niyaz, *et al.* [37]	2017	246	Empirical → Human labels
Zeng, Song, Yu, *et al.* [38]	2018	167	Empirical → Human labels

[5]Strictly speaking, the methodology used is empirical: it uses simulated trials to encode the analytic grasp quality metrics into the algorithm. However, encoding the analytic metrics into an empirical method is merely done to obtain run-time performance improvements, so the unanimous usage of analytic quality metrics remains an interesting trend

Table 2.6: Most influential publications using empirical approaches for grasping unknown objects, adapted from Bohg, Morales, Asfour, *et al.* [30] (2014) to match the vocabulary used in this thesis and include the publication years.

		Grasp quality attributes		Object features			Grasp quality metrics		
Author	Year	Local	Global	2D	3D	Multi-modal	Analytic	Human labels	Trials
Kraft, Pugeault, Baseski, *et al.* [39]	2008	X				X	X		
Popovic, Kootstra, Jorgensen, *et al.* [40]	2011	X				X	X		
Bone, Lambert, and Edwards [41]	2008	X			X		X		
Richtsfeld and Vincze [42]	2008	X			X		X		
Maitin-Shepard, Cusumano-Towner, Lei, *et al.* [43]	2010	X			X		X		
Hsiao, Chitta, Ciocarlie, *et al.* [44]	2010	X	X		X		X		
Brook, Ciocarlie, and Hsiao [45]	2011	X	X		X		X		
Bohg, Johnson-Roberson, Leon, *et al.* [46]	2011		X		X		X		
Stueckler, Steffens, Holz, *et al.* [47]	2011		X		X		X		
Klingbeil, Rao, Carpenter, *et al.* [48]	2011		X		X		X		
Maldonado, Klank, and Beetz [49]	2010		X		X		X		
Marton, Pangercic, Blodow, *et al.* [50]	2010		X		X		X		
Lippiello, Ruggiero, Siciliano, *et al.* [51]	2013		X		X		X		
Dune, Marchand, Collowet, *et al.* [52]	2008		X	X			X		
Kehoe, Berenson, and Goldberg [53]	2012		X	X			X		
Morales, Sanz, Pobil, *et al.* [54]	2006		X	X			X		

In contrast with empirical methods, analytic approaches to determine grasp quality scores provide traceability of the criteria that determined whether a certain grasp candidate was good or not. Analytic approaches are beneficial in cases like space robotics, where determinism is important and the consequences of a poorly executed grasp can be high. Furthermore, these provide a level of generalisation that allows grasps to be well identified even in the absence of prior knowledge of the target's shape. For these reasons, analytical grasp methods are the considered approach for this thesis work.

There are four primary analytical methods for grasp analysis, namely:

- **Spatial histogram features**: histograms applied to image recognition tasks which, instead of evaluating the distribution of the image as a whole, they divide it and evaluate the patches separately. This division allows for the preservation of some local information in the images, which can provide insight into the graspability of a certain area in multiple ways. Popular features for graspability analysis are tabulated in Table 2.7;

- **Geometric shape matching**: the grasping candidate can be geometrically evaluated by determining whether the bodies are unacceptably colliding against each other and whether the gripper contact points are in contact with the target body. This evaluation is depicted in Figure 2.16. To quantitatively assess the suitability of the grasp candidate, the voxels are assigned a score. For example, $\mathbf{V_o} \in [0 \rightarrow \mathbf{empty}, 1 \rightarrow \mathbf{occupied}]$ and $\mathbf{V_g} \in [-255 \rightarrow \mathbf{constraint}, 0 \rightarrow \mathbf{empty}, 1 \rightarrow \mathbf{contact\ point}]$, where $\mathbf{V_o}$ and $\mathbf{V_g}$ correspond to the object and gripper voxel grid, respectively. Then, the graspability score is simply the sum of the individual scores.

- **Probability of force/form closure**: force and form closure are the two ways to perform restraint analysis. Force closure uses friction force to achieve object closure, and it is achieved if for any non-contact wrench experienced by the object, contact wrench intensities exist that satisfy Equation 2.4. In other words, force closure is satisfied if and only if Equation 2.5 is met [55]. For 3D objects only two contacts are needed if they are modelled as soft fingers, and three non-collinear contacts if they are modelled as hard fingers. Form closure grasps are inherently also force closure and they describe how the object cannot move at all; always assuming hard fingers. They are formally described by the implication presented in Equation 2.6, which shows that if and only if a zero object displacement yields a gap function equal or larger than 0, then the grasp is first-order form closed [55]. It can be determined that in order to achieve form closure 7 contacts are required for 3D objects, except when different orders of form closure exist that leverage the shape of the objects and require a lower contact point count (Figure 2.17 demonstrates the workings of form closure at different orders).

- **Shape segmentation into primitive forms with pre-defined graspability metrics**: it is also possible to determine grasp quality by segmenting the object into primitive shapes. These primitives shapes can be obtained from point clouds by various methods, such as shape segmentation via recurrent neural networks [56]. The segmented shapes commonly consist of boxes, spheres, cylinders, and cones [34]. These primitive shapes possess predetermined grasp poses, as depicted in Figure 2.18.

$$\begin{aligned}\mathbf{M}_g(q)\ddot{q} + b_g(q,\dot{q}) + \mathbf{J}^T\lambda &= \tau_{app}\\ \mathbf{M}_o(u)\dot{\nu} + b_o(u,\nu) - \mathbf{G}\lambda &= g_{app}\end{aligned} \tag{2.4}$$

$$rank(\mathbf{G}) = n_\nu, \mathcal{N}(\mathbf{G}) \cap \mathcal{N}(\mathbf{J}^T) = 0 \tag{2.5}$$

$$\Psi(\bar{u} + du, \bar{q}) \leq 0 \Rightarrow du = 0 \tag{2.6}$$

Where:

- **M**$_{\{g,o\}}$ are symmetric, positive definite inertia matrices of the gripper and object, respectively.
- λ is the vector containing the vectors of the contact force and moment components transmitted through the contacts and expressed in the contact frames. It is expressed as $\lambda = \left[\lambda_1^T \dots \lambda_{n_c}^T\right]^T$ where n_c is the number of contact points, and $\lambda_i = \mathbf{H_i}\left[\begin{array}{cccccc} f_{in} & f_{it} & f_{io} & m_{in} & m_{it} & m_{io}\end{array}\right]$.
- **q** is the configuration of the gripper contact point.
- **b**$_{\{g,o\}}$ is the vector-product term.
- **J** is the gripper Jacobian.
- τ_{app} is the vector of external loads and actuator actions.
- **u** is the configuration of the object contact point.
- ν is the components of the twist.
- **G** is the grasp matrix that maps the object twist from the normal to the contact frame.
- **g**$_{app}$ is all applied wrenches (force and moment).
- $\Psi(\bar{u}, \bar{q})$ is the gap function vector that describes the physical contact between the object and the gripper contact points. It is equal to zero (0) if the contact points are touching, -1 if the contact is penetrated, and +1 if there is a gap between the contact points.

Table 2.7: Collection of linear and non-linear features relevant to spatial histogram grasp quality analysis.

Feature	Linearity	Rationale	Approach	Ref.
Law's Masks	Linear	Texture recognition for scene segmentation	Mask $(5 \times 5px)$	[57]
Local binary pattern	Linear	Texture-based encoder popular for object recognition	Mask $(3 \times 3px)$	[58]
Averaging colour filters	Linear	Colour uniformity often infers object continuity	Mask $(5 \times 5px)$	[36]
Oriented edge filters	Linear	Edges infer object discontinuities, which can learn to detect certain shapes	Mask $(5 \times 5px)$	[59]
Correlative depth	Non-linear	Hill-like depth profile of the grasping rectangle correlates to grasp quality	$\frac{\bar{d}_1\bar{d}_3}{\bar{d}_2{}^2} \neq 1$	[36]

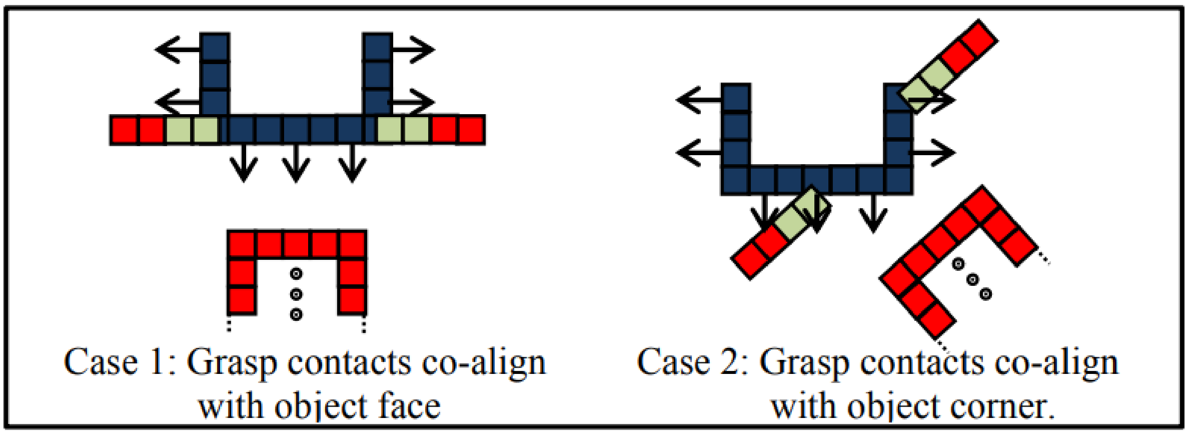

Case 1: Grasp contacts co-align with object face
Case 2: Grasp contacts co-align with object corner.

Figure 2.16: Example grasp assessment with the voxel grid representation. Depicted are gripper contact locations (light green), target object (blue), and Gripper body (red). Image obtained from Hegedus, Gupta, and Mehrandezh [60].

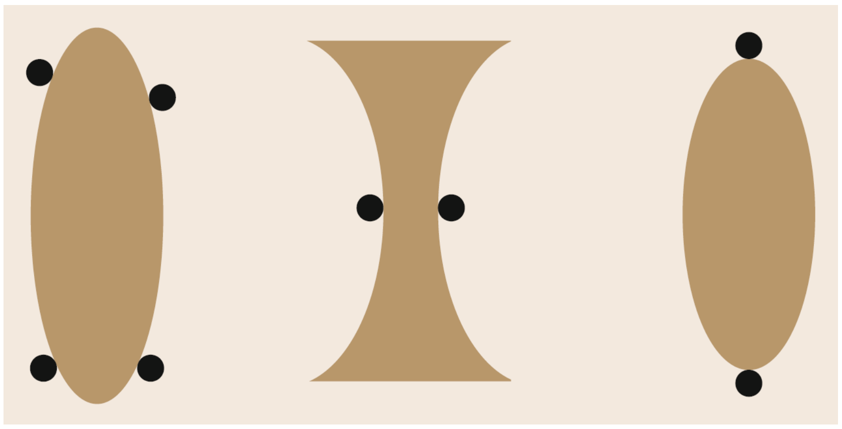


Figure 2.17: Three planar grasps. The rightmost does not achieve form closure and the other two achieve closure at different orders. Image obtained from Siciliano and Khatib [55].

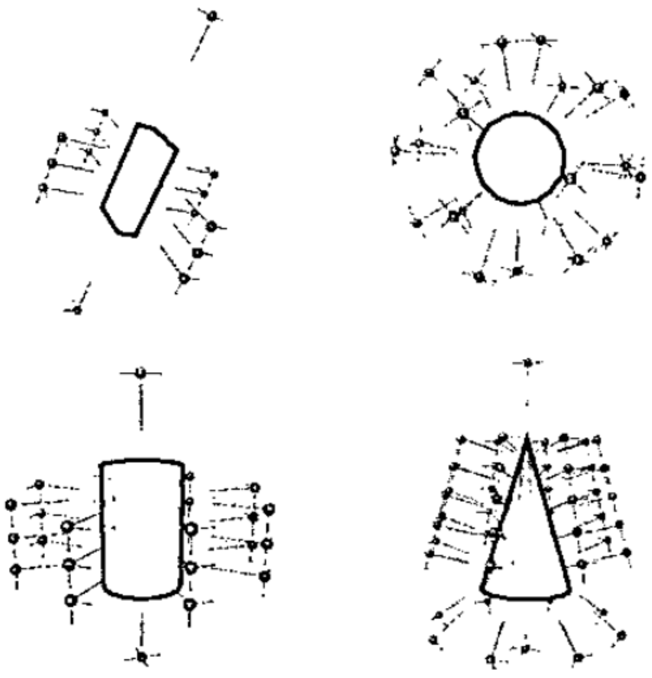


Figure 2.18: Examples of grasps generated from four different grasp primitive shapes, from Miller, Knoop, Christensen, *et al.* [34].

2.3.3. Grasp Candidate Generation

To generate candidates from a 3D map, an obvious solution would be using brute force: iterating through every voxel with a number of different orientations. However, this has a time complexity of $O(n + 3k)$ due to the 6D gripper configuration $(x, y, z, \alpha, \beta, \gamma)$. Furthermore, analysing every possible grasp pose throughout the surface is likely to be unnecessary as there likely are many regions where a grasp is not possible due to surface flatness or presence of fragile thermal shielding layer.

For deep learning-based grasp planning engines, the grasp candidates tend to be generated randomly in a sufficiently dense manner as to guarantee capturing successful grasp candidates (if any exist) [61], [62]. Nevertheless, deep learning works tend to rely on grasp inference from depth images, so their sampled grasping orientation is constrained to be aligned with the camera axis.

For the considered application in this thesis work, two optimisations to the instantiation of grasp candidates are contemplated:

- Neglect regions with high flatness and/or presence of thermal shielding layer: can be achieved by convoluting a broad edge detection mask on the depth channel to detect depth changes in the broad neighbourhood and with a tuned Laws mask on the colour channels to flag the space blanket texture;

- Fix the β and γ orientation of the grasp by analysing the surface normal at the center point such that only a 4D analysis is performed at every voxel. This reduces the time complexity to $O(n + k)$ while filtering out the least likely grasp candidates [63].

2.4. Improvement Gaps for Space Applications

Most of the previous works focus on obtaining a high yield of grasped targets over highest success probability at the expense of additional computational time. They also tend to use depth images as they are the most convenient representation for neural networks and also because static robots can only generally observe objects from a single view point (usually top-down). Creating a point cloud from single view depth images would be unprofitable as there would be many occlusions.

However, with the possibility of obtaining observations of the target object from multiple viewpoints in space due to the unconstrained dynamics of the systems, occlusion-free point clouds can be generated. Complete point cloud representations are advantageous spatial representations as they are well suited for geometric reasoning. With complete point clouds, for example, antipodal configurations of grasps can be readily analysed. These are a necessary condition for grasp success, as they provide information about scale compatibility. With the addition of extra analyses, such as surface contact normals, shape compatibility of the grasp can also be assured. A solution providing scale and shape matching is considered sufficient for achieving a good grasp.

Therefore, in space, analytic grasp planning metrics solely reliant on scale and shape geometric fitting are hypothesised to be sufficient for determining grasp quality. Furthermore, such analytic grasp planning metrics are well suited for global graspability analyses due to their usage of simple local surface properties and convolution can be applied between the target and gripper models to efficiently determine the grasp quality of all target surface data points. Ambitiously, it is hypothesised that including a grasp quality attribute to all target surface data points can enable complex graspability tasks in space, such as crawling with the robotic arms around the target surface to avoid propellant usage, or performing multiple grasps in different regions. These applications draw clear parallels to how traversability maps can enable autonomously-optimised point-to-point navigation for systems such the upcoming ESA Rosalind Franklin “ExoMars” rover, where autonomy is advantageous due to the significant communication delays.

II

Framework Design

3

Concepts and Architecture

This chapter focuses on generating concepts and the architecture of the system. It has a strong focus on systems engineering techniques that enrich the knowledge about the project, as interfaces are arguably single largest source of innovation in a project—as well as also being the least forgiving. In Figure 3.1, the classic systems engineering V-model representation for this thesis work is depicted. Throughout this thesis system elements and details will continuously be introduced as we approach the fully decomposed system state. At every step that is possible, parallels to the rationale and verification process will be introduced. This chapter begins with a definition of the mission needs and system environments, followed by stakeholder requirement generation that takes into account the legacy context of the facility in which the validation tests will be carried out. Then, the key performance parameters and acceptance criteria for the Aut-O-MAGIC project are presented, followed by the concept of operation, which presents of how the system is envisioned to interface with existing space technology and within its own subsystems. Lastly, insights into the system are drawn from the processes developed in this chapter.

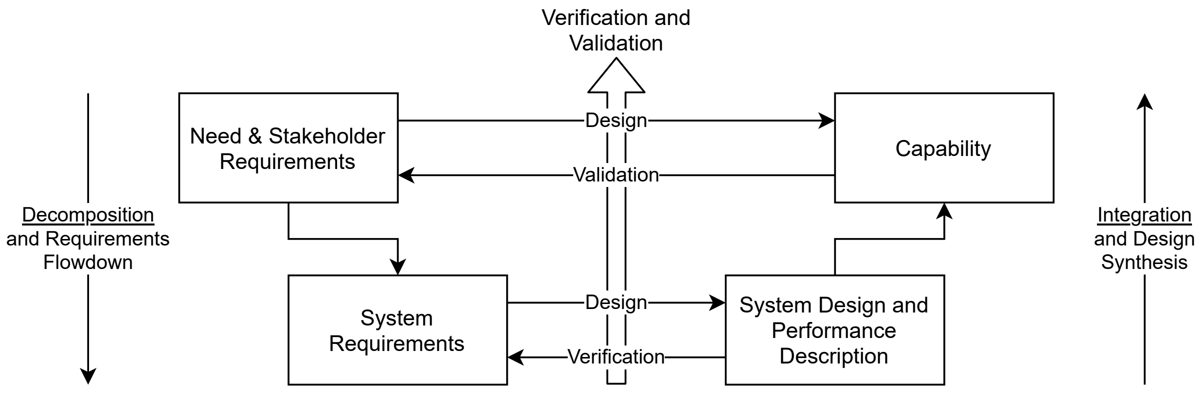

Verification and Validation
Need & Stakeholder Requirements
Design
Validation
Capability
Decomposition and Requirements Flowdown
Integration and Design Synthesis
System Requirements
Design
Verification
System Design and Performance Description

Figure 3.1: Systems Engineering V-Model for Autonomous On-Orbit Manipulability Analyser for Gripper Interaction Capabilities, emphasising the dual components of decomposition and integration in the product development process.

3.1. Mission Needs and System Environments

With the intention to define and scope the project expectations and requirements, the need and mission statement, as well as the system environments are developed.

3.1.1. Need and Mission statement

The customer expectations, which encompass the capability development desired from this thesis work, are in response to a functional deficiency in space operations. These expectations can be expressed as a mission statement, which represent what is to be achieved in opportunity space. Similarily, the research goals presented in section 1.3 also describe expectations, but do so in the knowledge space. Both expectation domains are of importance in the development of a novel system like the one proposed in this thesis.

The mission statement is formulated from a need statement that describes the functional deficiency that is to be addressed. The need statement for this work is as follows:

> Space missions have a growing need to interact with their environment without substantial ground support interaction and purpose-designed interfaces.

Then, the mission statement can be derived as follows:

> Aut-O-MAGIC will provide an autonomous grasp planning pipeline for robotic manipulation of arbitrary space objects, making available the surface-body understanding necessary for complex interaction of objects in space.

3.1.2. System Environments

The system is to be a technology demonstrator for novel grasp planning capabilities in space. As such, the most critical environmental elements are the ones that can be directly measured by the system's sensors. These include:

- Visual field: the on-board camera operates in the visual-spectrum. This environment can be simulated using Software. For validation tests, a space visual environment can be recreated by room darkening, a single static light source, and a (scaled) target object surface model.

- Physical space: depth information is obtained from the depth camera. This environment can be simulated using Software. Recreating the space physical space for validation tests requires an empty workspace area and a (scaled) target object surface model.

- Kinematics: kinematics are approximated by the state estimator in the system, which is part of the SLAM algorithm. This is necessary to correctly generate a 3D representation of the environment via the depth camera. For validation tests, physically recreating the relative motion of a typical target-chaser pair requires following a close proximity rendezvous trajectory and minimising vibrations introduced in the system as a consequence of the mechanical actuator system that executes the relative motion.

Other environmental elements, such as atmospheric conditions, do not play a significant role in the validation of this system as they do not vary considerably. It is worth noting that the facility presented in subsection 7.1.1, which will be used for system validation, is kept at a temperature of 21°C.

Summarising, the basic technological components of the system being designed are to be integrated with reasonably realistic supporting elements so they can be tested in both a simulated and laboratory environment relevant to the true operational environment.

3.2. Stakeholder Requirements

The project stakeholders are identified in Table 3.1. Their expectations vary depending on their type of involvement with the project, but can be largely classified as supporters of the functional need to be achieved with this project, supporters of knowledge to be gained, and entities influenced by the execution of the design work or future on-orbit operation. This section is divided in two subsections, that cover the legacy context of the facility used for the validation tests, and the stakeholder requirements. The legacy context is relevant to requirement generation because it exposes external constraints that the system being design must take into account in order to successfully interface with the validation facility.

Table 3.1: Stakeholders and their roles for the Aut-O-MAGIC project.

Stakeholder	Type
European Space Agency	Sponsor, Active
TU Delft	Sponsor, Passive
GRALS facility users	Passive
Target spacecraft owner	Active
Space community	Passive

3.2.1. The Legacy Context for Validation Tests

Due to the anticipated usage of the existing GRALS facility for validation of the system being designed, we have to understand how to fit in the current operational environment and existing systems landscape. These influence stakeholder requirements. In order to successfully interface with the legacy system to perform validation tests, the legacy processes, resources, and limitations shall be understood.

Legacy Processes

In subsection 7.1.1, use cases for the GRALS facility are described. These use cases are supported by the following processes:

- Facility usage workflow (Figure 3.2)
- Simplified data exchange of the considered systems (Figure 3.3)

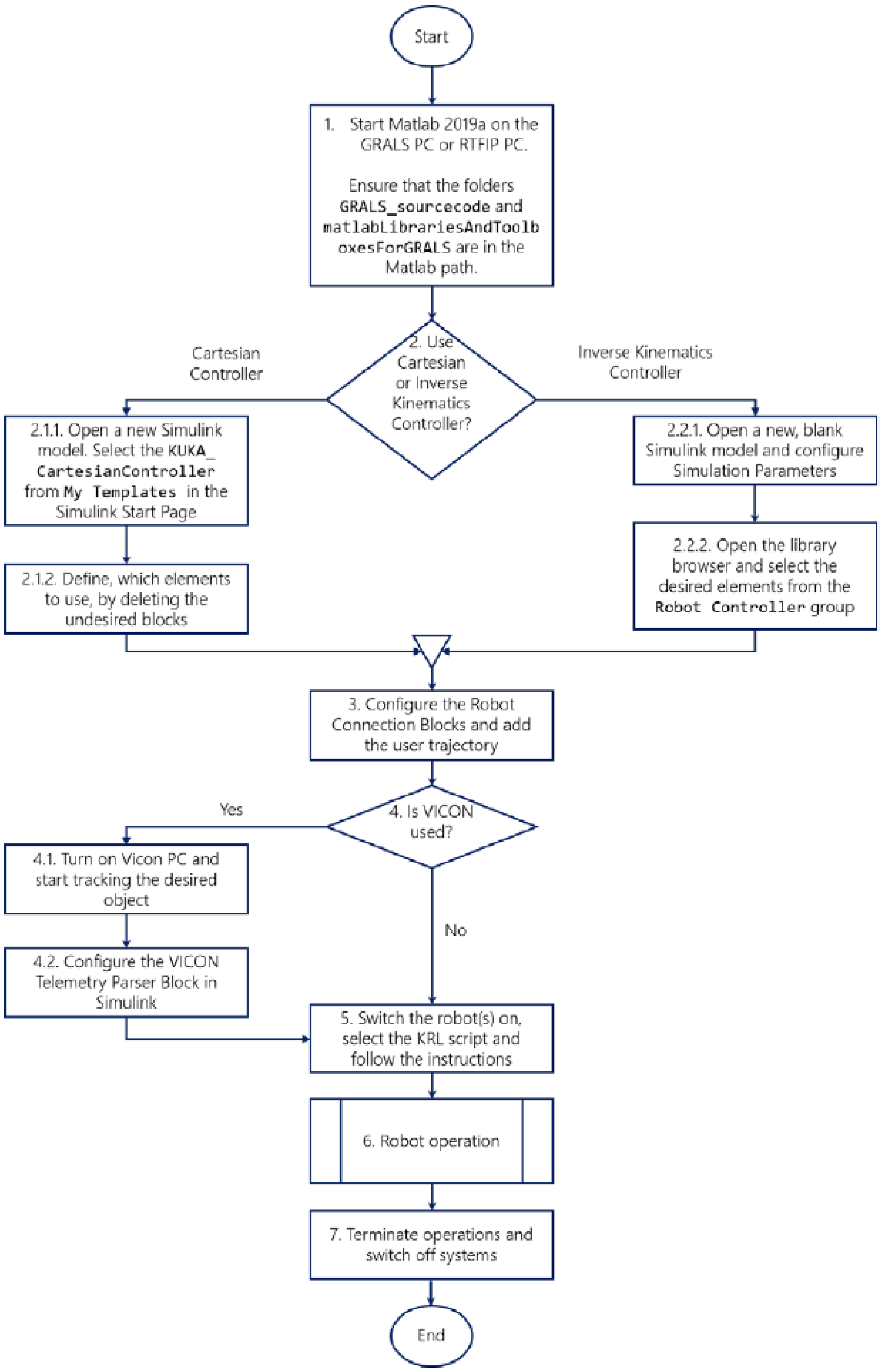

Start
1. Start Matlab 2019a on the GRALS PC or RTFIP PC.
Ensure that the folders GRALS_sourcecode and matlabLibrariesAndToolboxesForGRALS are in the Matlab path.
2. Use Cartesian or Inverse Kinematics Controller?
Cartesian Controller
Inverse Kinematics Controller
2.1.1. Open a new Simulink model. Select the KUKA_CartesianController from My Templates in the Simulink Start Page
2.1.2. Define, which elements to use, by deleting the undesired blocks
2.2.1. Open a new, blank Simulink model and configure Simulation Parameters
2.2.2. Open the library browser and select the desired elements from the Robot Controller group
3. Configure the Robot Connection Blocks and add the user trajectory
4. Is VICON used?
Yes
No
4.1. Turn on Vicon PC and start tracking the desired object
4.2. Configure the VICON Telemetry Parser Block in Simulink
5. Switch the robot(s) on, select the KRL script and follow the instructions
6. Robot operation
7. Terminate operations and switch off systems
End

Figure 3.2: Workflow on how to run an experiment with the GRALS facility, from Schwendener, Vilella, and Bänninger [64].

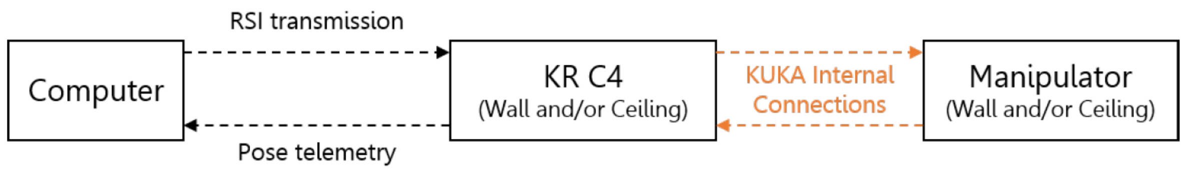


Figure 3.3: Simplified topology of the considered systems in GRALS, from Schwendener, Vilella, and Bänninger [65].

Legacy Resources
The legacy resources, both in terms of hardware and software development tools, used to perform the legacy processes described in the previous section, are as follows:

- Figure 3.4: facility hardware communication interfaces.
- Figure 3.5: robotic arms mechanical interface.
- Figure 3.6: facility functional principle using the legacy tools on the remote PC.

Limitations
As with anything, there are some limitations to take into account:

- Mechanical limitations
  - The payload capacity of the GRALS facility robotic arms is up to 10kg, depending on the payload's inertial distribution (see Figure 3.7).
- Safety limitations
  - The usable workspace volume is approximately $2000 \times 32000 \times 1500$ mm.
  - The payload can be transported at a speed of up to 250 mm/s and 30°/s, or up to 2000 mm/s with strict safety measures, formally trained operator, and proper justification.
- Electrical limitations
  - Power output limited to 24V/3A from robot connector or unlimited power with independent tether.
- Computational limitations
  - The RSI cycle time is 4ms.
  - Deploying the solution in Simulink requires that the code is compatible with code generation; such that it is compiled, instead of interpreted.
  - Aggregating all services in a single Simulink solution can lead to poor real-time performance. Simulink is a single-threaded application.

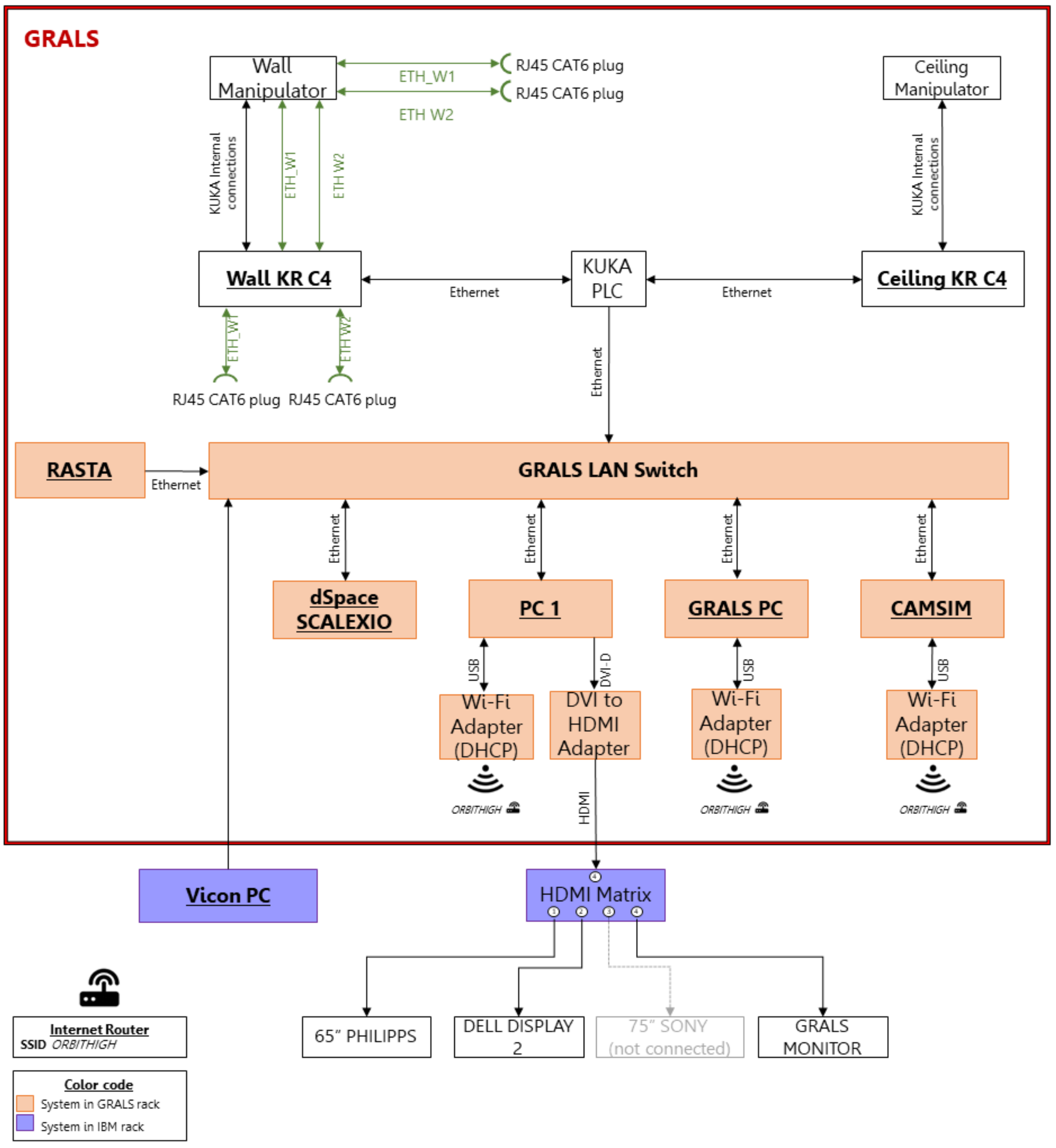


Figure 3.4: GRALS hardware connectivity (communications) diagram, where any device on the GRALS LAN can interface with the KUKA systems. From Schwendener, Vilella, and Bänninger [64].

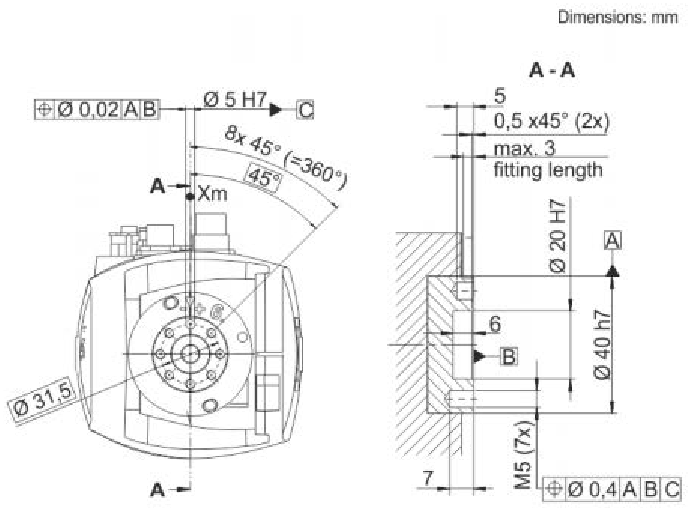


Figure 3.5: Mounting flange taken from the KR10 R1100 sixx C operating instructions. From KUKA Robotics Corporation [66].

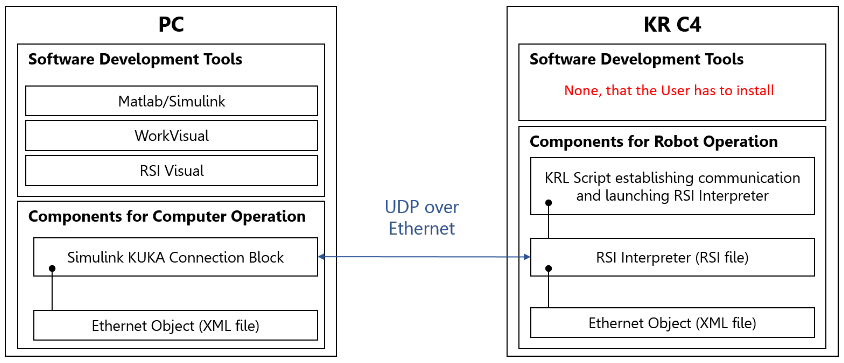

PC
Software Development Tools
Matlab/Simulink
WorkVisual
RSI Visual
Components for Computer Operation
Simulink KUKA Connection Block
Ethernet Object (XML file)
UDP over Ethernet
KR C4
Software Development Tools
None, that the User has to install
Components for Robot Operation
KRL Script establishing communication and launching RSI Interpreter
RSI Interpreter (RSI file)
Ethernet Object (XML file)

Figure 3.6: Functional principle of GRALS operation using a remote PC. From Schwendener, Vilella, and Bänninger [65].

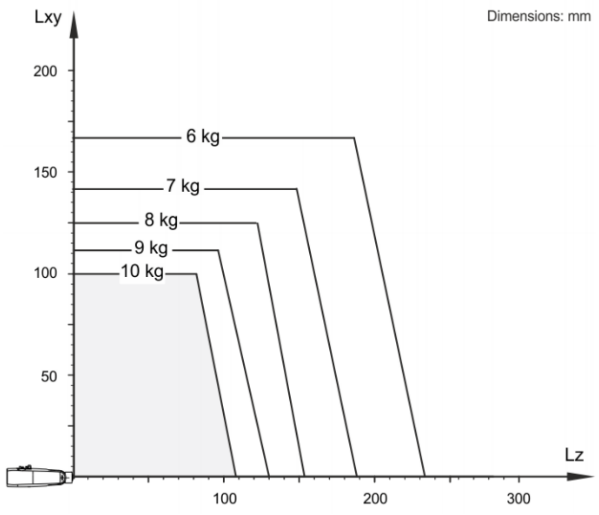


Figure 3.7: Maximum mass of payload, depending on its inertial distribution. From KUKA Robotics Corporation [66].

3.2.2. Stakeholder Requirements

The stakeholder requirements include general System requirements, labelled [STA-REQ-X]; and GRALS facility requirements to be taken into account for the physical integration of the system, labelled [STA-REQ-VAL-X].

[STA-REQ-1]	(Capability) The system shall enable autonomous grasp planning of arbitrary space objects. *Rationale*: This requirement captures the primary research objective of the thesis work.
[STA-REQ-2]	(Characteristic) The system shall not require any elements to be present in the target object. *Rationale*: The research focuses on grasp planning on arbitrary objects, which are not purposely fitted for grasping, so no elements can be expected in the target.
[STA-REQ-3]	(Characteristic) The system shall not generate space debris. *Rationale*: Space debris is becoming increasingly concerning and the stakeholder vision is to contribute to the clean-up of space debris, and by extension not generate more.
[STA-REQ-4]	(Capability) The system shall provide a global map of the graspability of the target space object. *Rationale*: This requirement captures a research objective of this thesis work, which is to generalise graspability in a way analogous to traversability cost maps in rover navigation, so a global map is essential.
[STA-REQ-5]	(Characteristic) The system shall use analytic grasp quality metrics. *Rationale*: The methodology decided for this thesis focus on geometric analyses using 3D maps, which are performed using analytic methods.
[STA-REQ-VAL-1]	(Characteristic) The system shall have mass properties within the range described in Figure 3.7. *Rationale*: The facility used for validation tests is constrained by this capacity.
[STA-REQ-VAL-2]	(Characteristic) The system shall use a mechanical interface compatible with the mounting flange of the KUKA KR10 R1100 sixx C. *Rationale*: The equipment in the facility used for validation tests requires using this specific mechanical interface.
[STA-REQ-VAL-3]	(Characteristic) The system shall use the RobotSerialInterface Ethernet-based communication protocol to interface with the GRALS facility. *Rationale*: The facility used for validation tests solely uses this communication protocol for data transport.

3.3. Key Performance Parameters and Acceptance Criteria

Key Performance Parameters (KPP) for the project are tabulated in Table 3.2, together with its performance drivers. The acceptance criteria, in Table 3.3, are then set based on the scope of this project and the dependencies of any other performance drivers.

Table 3.2: Aut-O-MAGIC Key Performance Parameters and Performance Drivers.

Key Performance Parameters (KPP)	Performance Drivers
Resolution — Reconstruct high fidelity 3D maps from the sensor input	Voxel resolution
Range — Operational range of the system	Camera resolution
Reliability — Minimise the grasp failure rate	Minimum acceptable grasp quality value
Repeatability — Obtain a high degree of agreement between results in different iterations	Robustness

Table 3.3: Aut-O-MAGIC Acceptance Criteria.

Criterion (KPP)	Value	Rationale
Resolution	2.5 cm	Given an ordinary gripper graspable area of $5 \times 3 \times 2$cm [62], a sampling of half the gripper width, 2.5 cm, would assure observation of sufficiently thin structures that the gripper could grasp.
Range	0.5–5 m	Given that the scope of this work is accurate, close range measurements, a camera system covering the 0.5–5 m range is ideal. This will allow precision comparisons at different distances to draw conclusions on the maximum allowable distance to resolve the target features at the specified acceptance resolution.
Reliability	99%	Qualitative performance driver. Space operations are safety-critical, and priority must be given to only attempting grasps that are of highest degree of confidence. In practise, this criterion is impossible to test without running extensive physical grasping tests, which are beyond the scope of this work.
Repeatability	Full	Solution shall follow a deterministic approach, which yields the exact same results every time.

3.4. Concept of Operations

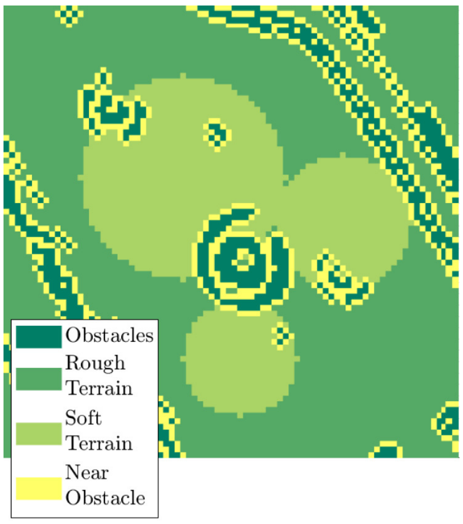
As described in subsection 3.1.1, the goal of this work is to provide an autonomous grasp planning pipeline for robotic manipulation of arbitrary space objects. This pipeline is envisioned to enable the execution of complex on-orbit manipulation tasks autonomously. Drawing parallels to path planning for unmanned vehicles, the grasp planning pipeline subject to design is to provide a high-level understanding of the graspability of any surface region in the target object, similar to a traversability cost map for rover path planning (see Figure 3.8). This high-level understanding of global graspability of the target object is necessary when considering manipulation tasks[1] such as on-orbit repairs, de-tumbling, and crawling traverses across the target object surface.

Overall, it is envisioned that the capabilities of Aut-O-MAGIC can help enable uncooperative rendezvous and docking, removing the need for a cooperative relative navigation system. In Figure 3.9, the current “as-is” operational environment of relative navigation is illustrated while, in Figure 3.10, the envisioned environment with autonomous uncooperative navigation is described.

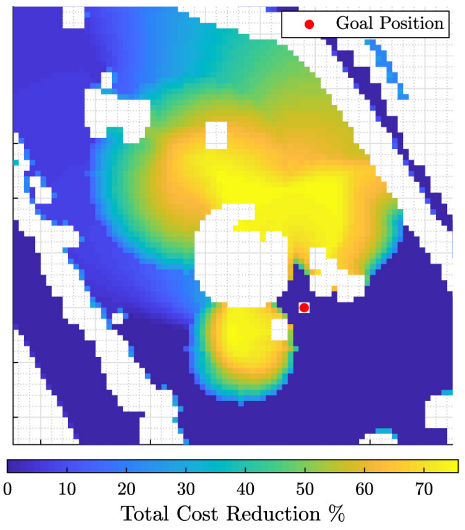
3.5. Conclusions

This chapter presented the need for a global grasp planning system such as the one proposed with Aut-O-MAGIC. It investigated the constraints imposed by external factors such as the validation tests, and devised stakeholder requirements to meet the mission goals and comply with external constraints. Then, key performance parameters for the system being designed have been elaborated, together with acceptance criteria that dictate the minimum performance the system shall display in order to perform satisfactorily. Lastly, visual representations of the concept of operations of the system within the space proximity operations environment is depicted, which displays a substitution of the cooperative relative navigation system in favour of a chaser-only perception system that compiles a target spacecraft model from sensor data. This envisioned concept of operations draws clear parallels with terrain traversability maps, as it readily provides global graspability information that is required for planning complex manipulability tasks that require more than a single gripper action.

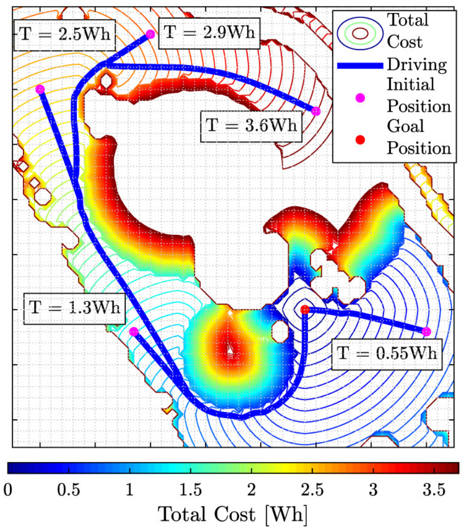
[1]Manipulation can be loosely defined as grasping over time; grasping multiple times at possibly different locations



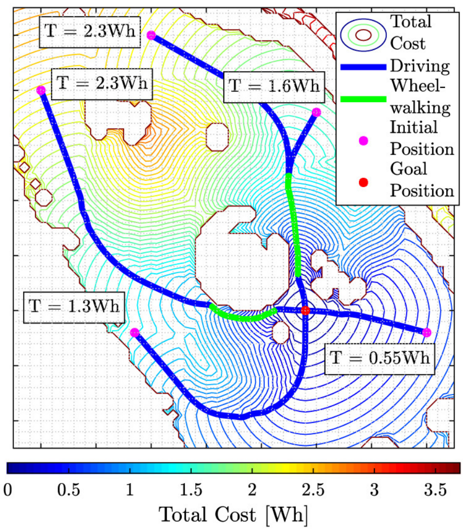
(a) Distribution of the different types of terrain over the *Global Layer*.



(b) Percentage of *Total Cost* reduced considering also *Wheel-walking*.



(c) *Global Path Planning* only considering *Normal driving*.



(d) *Global Path Planning* considering *Normal driving* and *Wheel-walking*.

Figure 3.8: Figures including the traversability cost maps for rover navigation in different terrains. The grasp planning pipeline being designed in the scope of this thesis is envisioned to be able to generate cost maps similar to figures (c) and (d), from which a global manipulation plan can be devised, similar to how a global path planning plan is devised from these figures. Figures obtained from Sánchez-Ibánez, Pérez-del-Pulgar, Azkarate, *et al.* [67]

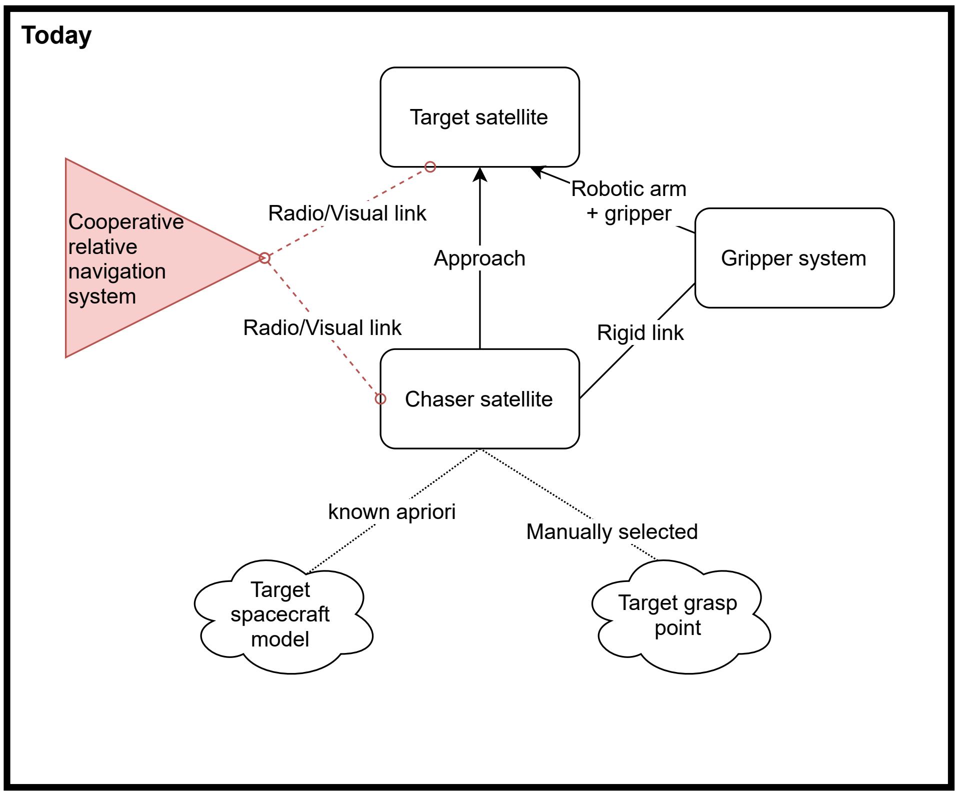


Figure 3.9: The 'as-is' Environment to be addressed by the Aut-O-MAGIC system, in red are the elements to be replaced.

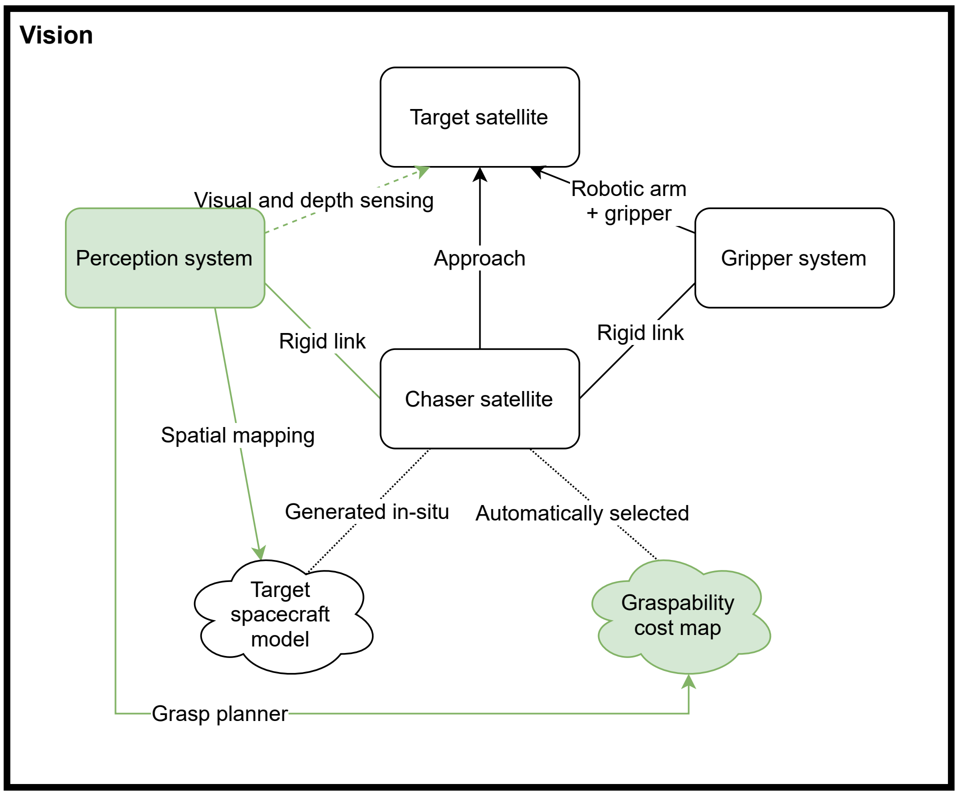


Figure 3.10: The envisioned Aut-O-MAGIC system environment, with the green elements substituting the cooperative navigation system block.

4 System Design

This chapter treats the design of the Aut-O-MAGIC system. First, the technical requirements are defined based on research objectives, stakeholder requirements, and inherited constrains from the acceptance criteria. This is followed by trade studies of the potential hardware and software programming frameworks to use. These are then selected, and the software system decomposed into blocks of single-responsibilities. This decomposition allows the generation of derived technical requirements, which further direct the system design implementation. Then, the overall design solution is defined, containing each of the software divisions drawn during decomposition. Lastly, the chapter is closed with conclusions on the outcomes of the systems design approach followed in this chapter.

4.1. Technical Requirements Definition

With the research objectives clear, the stakeholder requirements determined, and clear constrains inherited from the acceptance criteria and legacy context, the top-level technical requirements can be derived:

[SYS-REQ-1]	(Functional) The system shall generate a 3D reconstruction of the scene captured by the 3D sensor. *Rationale*: The global grasp planning premise is based on availability of 3D maps, which require reconstruction from a 3D sensor input. *Criticality*: High. *Verification method*: test.
[SYS-REQ-2]	(Functional) The system shall generate grasp candidates of the target object. *Rationale*: Grasp quality analysis requires candidates to analyse. *Criticality*: High. *Verification method*: analysis.
[SYS-REQ-3]	(Functional) The system shall rank grasp candidates based on an established measure of grasp quality. *Rationale*: Global grasp planning is based on assigning a score to each grasp candidate. *Criticality*: High. *Verification method*: test.
[SYS-REQ-4]	(Functional) The system shall generate a 3D reconstruction of a synthetic 3D model. *Rationale*: Analysing synthetic noise-free 3D models is a desired step before analysing real-world data, as it reduces the sources of error in the system. *Criticality*: Medium. *Verification method*: test.
[SYS-REQ-5]	(Functional) The system shall provide a visual representation of the target object grasp quality. *Rationale*: Grasp quality analyses are difficult to quantify and benchmark against alternative implementations because of non-standardised graspability scores. Visualisation methods are essential in verifying the fitness of a grasp candidate *Criticality*: High. *Verification method*: test.
[SYS-REQ-6]	(Functional) The system shall provide an interface for result comparison with depth image-based grasp planning solutions. *Rationale*: Comparison with other grasp planning methods is a good practise for establishing that the grasps suggested are sensible and in-line with current state-of-the-art techniques. *Criticality*: Medium. *Verification method*: test.
[SYS-REQ-7]	(Non-functional) The system must be able to generate a 3D model with a resolution of 2.5 cm *Rationale*: This is an acceptance criteria. It dictates the minimum resolution required to resolve geometric features for grasping *Criticality*: High. *Verification method*: inspection.
[SYS-REQ-8]	(Non-functional) The validation runs shall cover a range envelope of 0.5–5m. *Rationale*: This is an acceptance criteria. This range allows analyses at different distances to draw conclusions on the maximum possible distance to resolve the required target features, as well as the constraints in achieving the required resolution *Criticality*: Medium. *Verification method*: inspection.
[SYS-REQ-9]	(Non-functional) The system shall instantiate grasps in a deterministic fashion. *Rationale*: It is desirable to instantiate grasps in a way through which repeatable analyses can be performed. *Criticality*: Low. *Verification method*: analysis.

4.2. Trade Studies

In the system design, two trade studies are performed to determine the optimal system configuration for this work. These are:

- 3D imaging hardware component selection for system validation;
- Software programming framework to use.

4.2.1. 3D Imaging Hardware

As described in subsection 2.1.2, different sensors types can provide 3D data. However, only camera-type sensors provide an accuracy below 0.01 m (Figure 2.3). Under normal circumstances, a morphological matrix would be developed to find the ideal camera parameter combination for the task at hand, and then move onto purchase selection. However, in this work, only consumer COTS products are considered. Together with the preference of hardware that requires minimum time to begin operating, only a handful of products remain candidates.

For the purpose of the real-world data collection presented in chapter 7, three state-of-the-art camera-type sensors are considered for this work. Their properties are tabulated in Table 4.1.

From the specifications table and the provided sensor data samples, it can be seen how the Microsoft Azure Kinect DK (K4A) is the most versatile 3D imaging solution. Both the K4A and the Intel RealSense D455 RGB-D cameras provide less noisy captures than the Intel RealSense L515 LIDAR solution (Figure 4.1, Figure 4.2), while the K4A also outperforms the D455 in FOV, short-range operation, and RGB camera resolution.

Table 4.1: State-of-the-art camera-type 3D sensors specification trade-off. Some of the camera parameters require a trade-off, i.e. better resolution at lower FPS or FOV; the specifications shown are best individual values. *D* stands for depth sensor, and *C* for colour (RGB) sensor.

Sensor	Type	Resolution (px)	Range (m)	FOV (°)	FPS	Others
Intel RealSense L515	LiDAR + RGB	D:1024×768 C:1920×1080	$0.25 - 9$	D:70×55 C:70×43	30	Incl. IMU
Microsoft Azure Kinect DK	RGB-D	D:1024×1024 C:4096×3072	$0.2 - 5.5$	D:120×120 C:90×74.3	30	Incl. IMU, 7 microphone array
Intel RealSense D455	RGB-D	D:1280×720 C:1280×800	$0.4 - 6$	D:86×57 C:86×57	D:90 C:30	Incl. IMU

[1]https://youtu.be/RBDfABrNKz8
[2]https://youtu.be/RtqfdCUIUrM

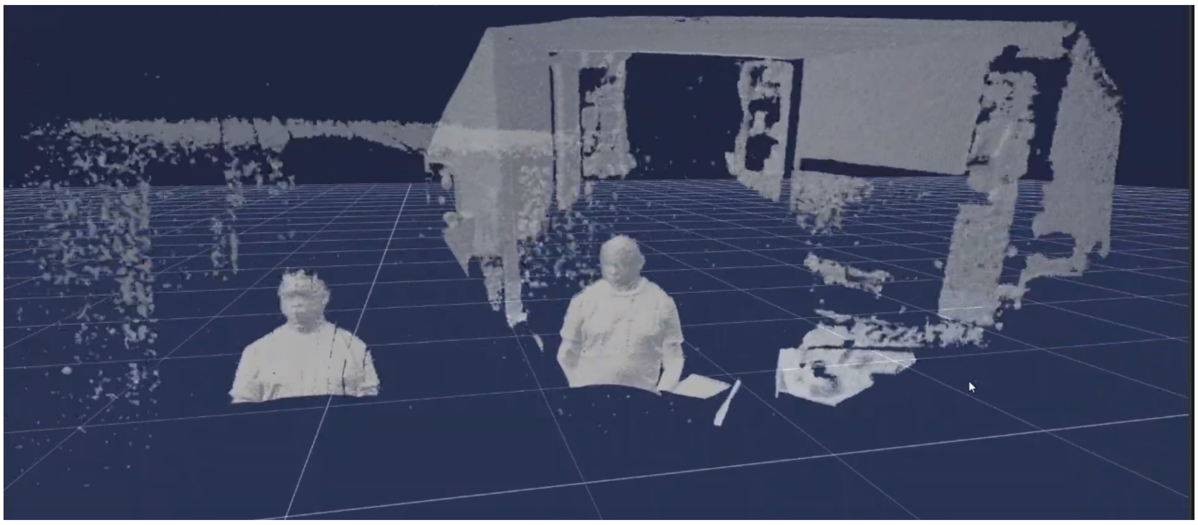


Figure 4.1: Pointclouds in Intel RealSense L515 (left) vs Microsoft Azure Kinect DK (right), from brekel.com on YouTube[1].

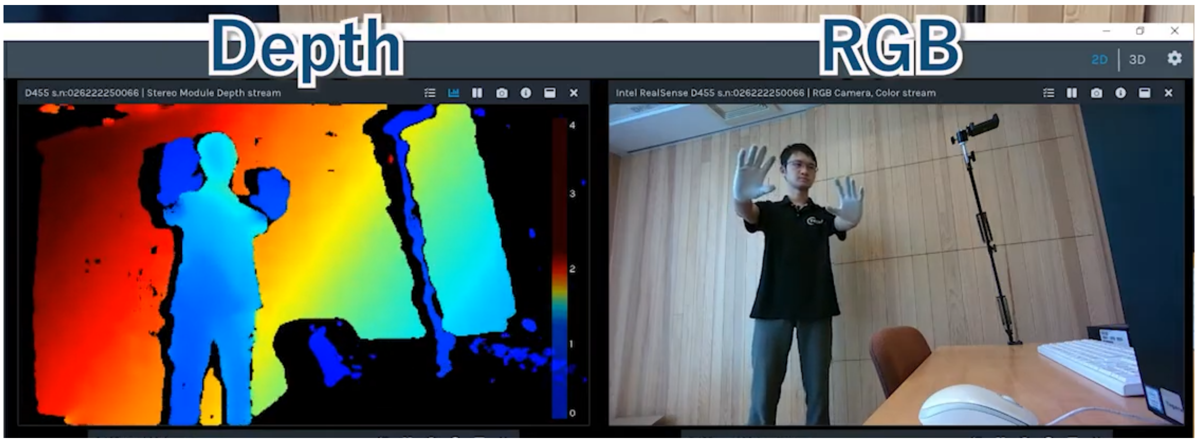

Depth
RGB
2D
3D
D455 s.n:026222250066 | Stereo Module Depth stream
Intel RealSense D455 s.n:026222250066 | RGB Camera, Color stream
4
3
2
1
0

Figure 4.2: 2D depth map and RGB image in Intel RealSense D455, from Tegara Corporation on YouTube[2].

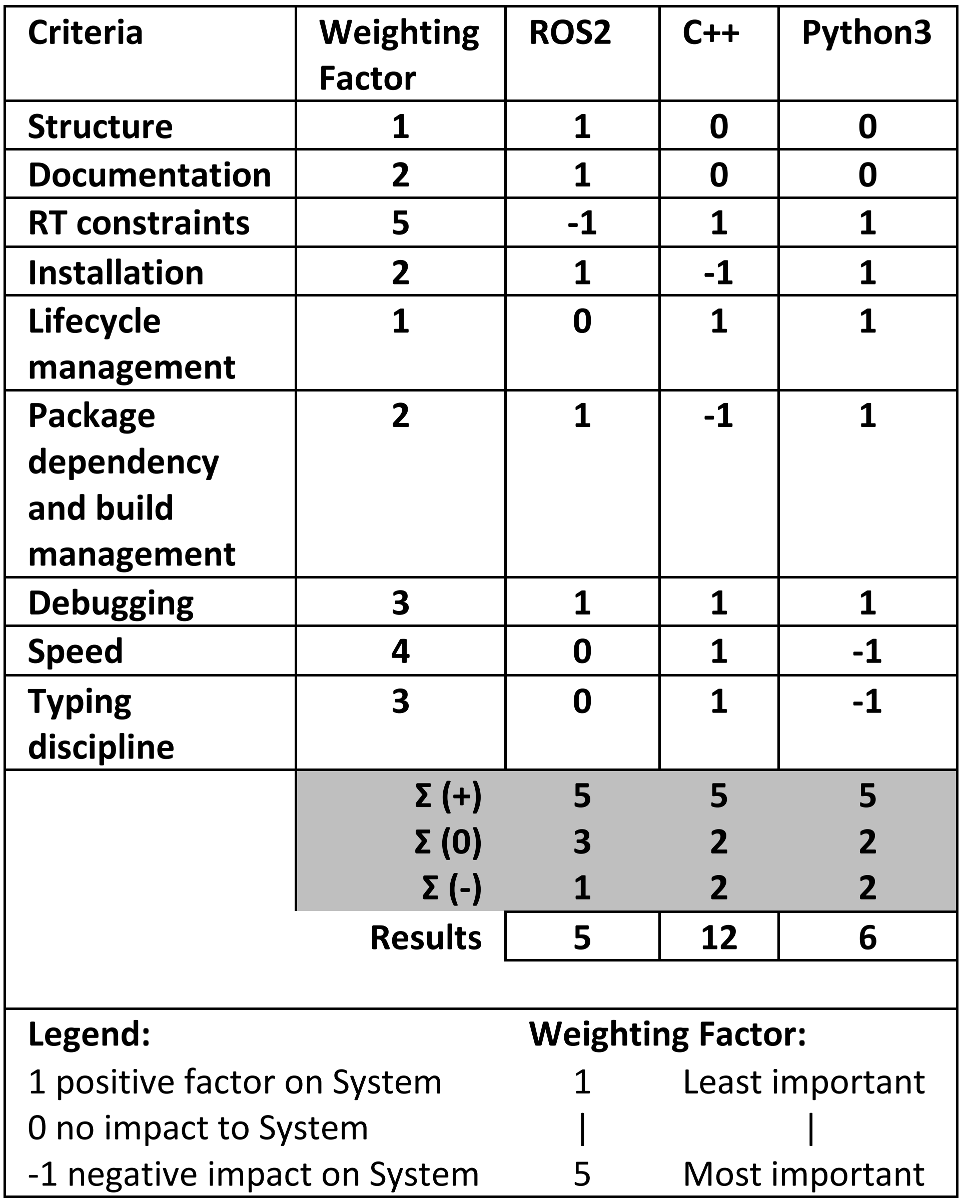
4.2.2. Software Programming Framework

It is beneficial to resort to pre-existing software programming frameworks where appropriate to minimise development effort. In essence, we see two main areas where some design could be abstracted away to a pre-existing software framework:

- Communication between software components;
- 3D data representation and operation.

Regarding the software communication framework, three options exist: ROS2 (via either C++ or Python3), pure C++, or pure Python3. Given the modularity of the software systems used in this work due to their multi-disciplinary nature, interaction between them is foreseen to be necessary frequently. OctoMap provides interfaces for all three communication frameworks, so it largely depends on a trade-off between overhead and benefits, which is summarised in Figure 4.3, with detailed reasoning below.

- ROS2 provides a structured development approach with industry in mind, while C++/Python do not provide any initial structure.
- ROS2 provides extensive documentation in the usage of their framework, while C++/Python rely on Q/A forums and libraries documentation.
- ROS2 is targeted towards robotic real-time systems, and hence it preferably works with *streams* of data that may get dropped or queued if not processed at speed. On the other hand, C++/Python iteratively process data at their own pace without additional configuration.
- ROS2 and Python3 are very easy to install and run, being available from the `apt` repository and having convenient launch methods. On the other hand, C++ relies on offline compilers that need to be installed and CMake for building the software, which is a learning curve.
- Lifecycle management: ROS2 provides an interface for state transitions via launch files, which can prove useful if the program shall be able to selectively execute different non-related operations. On the other hand, C++ and Python provide good and simple CLI interface options and arguments that can further increase the versatility of the solution with a decreased investment.
- Package dependency and build management: ROS2 and Python provide generally straight-forward methods for package management (ROS2: ament_cmake, colcon, rosdep; Python: pip), while C++ provides no such package management system and requires libraries to be manually linked against the targets via CMake.
- Debugging: ROS2 and C++ rely on command line debugging methods. ROS2 has some abstracted CLI utilities such as `ros2 node` and `ros2 topic`, as well as `rqt tools` that make it easier to see running processes. On the other hand, due to the interpreted nature of (C)Python, it supports breakpoints for streamlined debugging efforts. The GNU Debugger (GDB) is a practical tool for debugging code in C++.



Criteria	Weighting Factor	ROS2	C++	Python3
Structure	**1**	**1**	**0**	**0**
Documentation	**2**	**1**	**0**	**0**
RT constraints	**5**	**-1**	**1**	**1**
Installation	**2**	**1**	**-1**	**1**
Lifecycle management	**1**	**0**	**1**	**1**
Package dependency and build management	**2**	**1**	**-1**	**1**
Debugging	**3**	**1**	**1**	**1**
Speed	**4**	**0**	**1**	**-1**
Typing discipline	**3**	**0**	**1**	**-1**
	Σ (+)	**5**	**5**	**5**
	Σ (0)	**3**	**2**	**2**
	Σ (-)	**1**	**2**	**2**
	Results	**5**	**12**	**6**

Legend:	**Weighting Factor:**	
1 positive factor on System	1	Least important
0 no impact to System	|	|
-1 negative impact on System	5	Most important

Figure 4.3: Pugh Matrix for the software communication framework.

- Speed: While ROS2 can run using Python or C++ nodes, pure C++ is still faster due to lack of abstractions in communication. Without a doubt, the slowest is Python due to its interpreted nature.

- Typing discipline: Statically-typed languages such as C++ allow much more extensive error-checking at compile-time or at rest. Python, on the other hand, delegates significant error checking to run-time. Compile-time error checking aids minimise the bugs in code and time spent debugging.

For 3D data representation, the options are OctoMap, PointCloud Library (PCL), and Open3D. While they all provide methods for operating on voxels and point clouds, OctoMap is exceptionally designed for derived class generation.

4.3. Hardware Component Selection

The chosen time-of-flight camera was the Microsoft Azure Kinect DK, depicted in Figure 4.4, with the below specifications:

- Range: 0.2–5.50 m;

- Up to 30FPS;

- Resolution (at 15FPS): 1024×1024px depth, 4096×3072px RGB;

- Resolution (at 30FPS): 640×576px NFOV 512×512px WFOV depth, 3840×2160px RGB;

- FOV:
  - Depth sensor: 75° × 65° NFOV, 120° × 120° WFOV;
  - RGB sensor: 90° × 74.3° and 90° × 59°;

- Including IMU and 7-microphone array.

Figure 4.4: The Microsoft Azure Kinect DK time-of-flight camera. Credits: Microsoft Corporation.

4.4. Software Component Decomposition

The logical software component decomposition is the process for creating detailed functional software requirements to meet the expectations for this work. A system architecture model is created that reflects the functional organisation of the system at element-level. This enables relationships, dependencies, and interfaces to be clearly defined and used to generate derived technical requirements.

4.4.1. System Architecture Model

The system architecture model for the Aut-O-MAGIC project is depicted in Figure 4.5, it demonstrates the need to decompose the software solution into the following components, in order to meet the single-responsibility principle:

- Sensor model acquisition: to process the data captured by the 3D sensor and compile it into a global 3D map.
- 3D model voxeliser: to process a synthetic 3D model into a global 3D map.
- Grasp Planning Engine: to analyse the grasp quality of a 3D model.
- 3D model to depth map generator: to project a 3D map into depth images from a user-defined viewpoint for easy benchmarking against other state-of-the-art grasp planning solutions that rely on depth images.

One note to make is that throughout this report we refer to voxel trees with a binary occupancy attribute as *binary trees*, whereas another common name for them, used by other work, is *bonsai trees*. The name octrees is used for trees with arbitrary attributes such as graspability score, probability-based occupancy, etc.

4.4.2. Derived Technical Requirements

From the system architecture model developed, we are able to further decompose the software technical requirements with the additions found below. Their addition is based on pure technical decomposition of the system architecture model devised, such that their rationale is always in-line with their parent requirements and their criticality coupled with each other's successful completion. The requirements are appended with the following identifiers to label each of the subsystems:

- SA: Sensor model acquisition;
- VOX: 3D model voxeliser;
- GP: Grasp planning engine;
- DEPTH: 3D model to depth map.

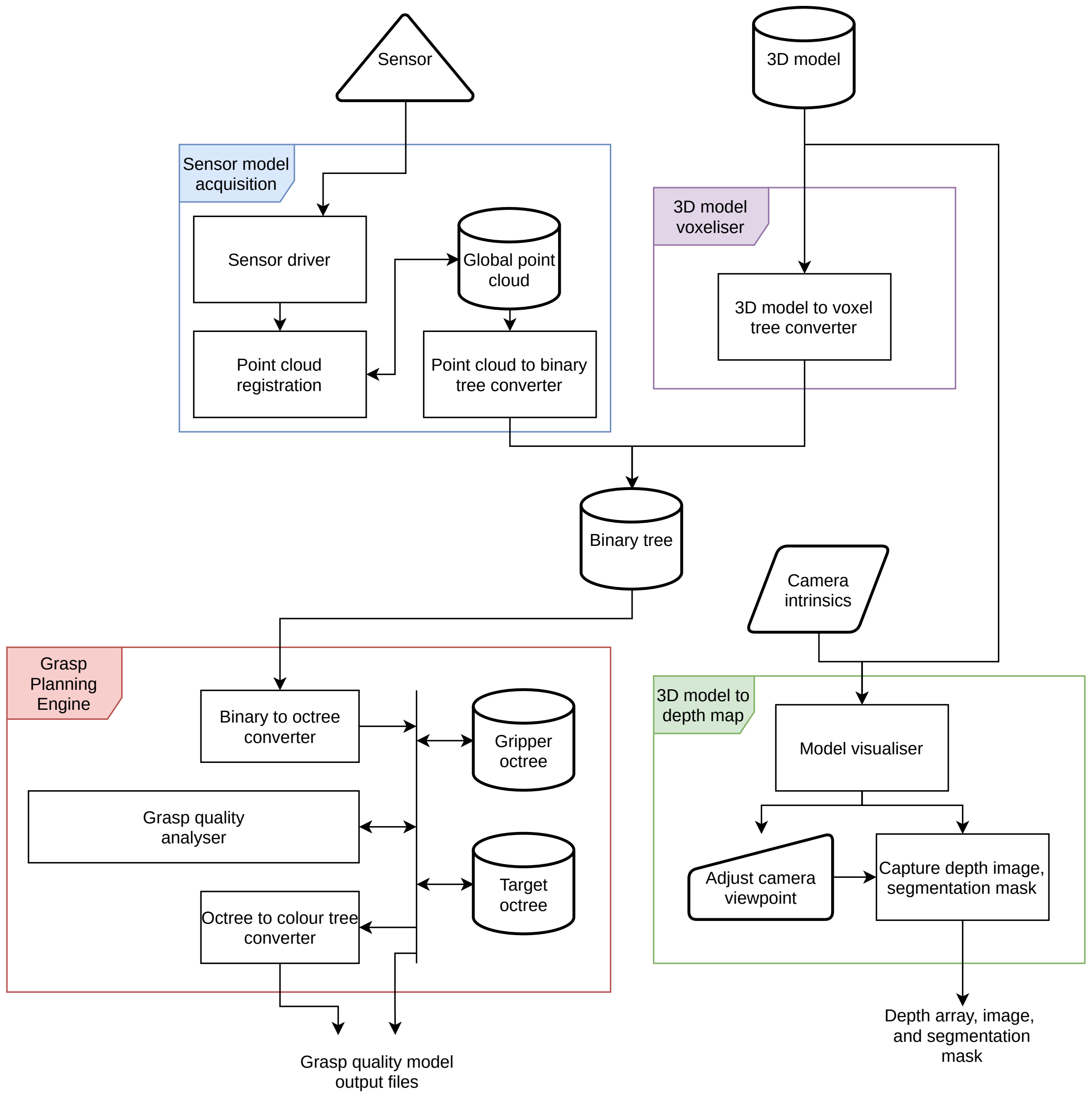


Figure 4.5: Software system architecture model for the Aut-O-MAGIC project.

[SW-REQ-SA-1]	(Functional) The system shall provide a driver for the sensor that makes available its observed point cloud. *Verification method*: test.
[SW-REQ-SA-2]	(Functional) The system shall be capable of registering sensor point cloud snapshots against an existing map. *Verification method*: test.
[SW-REQ-SA-3]	(Functional) The system shall compile a time-invariant global point cloud map. *Verification method*: test.
[SW-REQ-SA-4]	(Functional) The system shall be able to convert a point cloud to a binary voxel tree. *Verification method*: test.
[SW-REQ-VOX-1]	(Functional) The system shall be able to convert a 3D mesh model to a binary voxel tree. *Verification method*: test.
[SW-REQ-GP-1]	(Functional) The system shall be able to convert a binary voxel tree to an octree with custom grasp planning node attributes. *Verification method*: test.
[SW-REQ-GP-2]	(Functional) The system shall map the grasp quality of the octree nodes into attributes within those nodes. *Verification method*: test.
[SW-REQ-GP-3]	(Functional) The system shall iteratively analyse the grasp quality of all nodes in the octree. *Verification method*: analysis.
[SW-REQ-GP-4]	(Functional) The system shall provide an octree class definition for a gripper octree model. *Verification method*: inspection.
[SW-REQ-GP-5]	(Functional) The system shall provide an octree class definition for a target octree model. *Verification method*: inspection.
[SW-REQ-GP-6]	(Functional) The system shall convert the processed octrees into colour trees with the grasp quality mapped into the colours. *Verification method*: inspection.
[SW-REQ-GP-7]	(Functional) The system shall write to file the processed octree models at the end of execution. *Verification method*: test.
[SW-REQ-DEPTH-1]	(Functional) The system shall create a virtual camera of the provided 3D mesh model based on user-defined camera intrinsics. *Verification method*: test.
[SW-REQ-DEPTH-2]	(Functional) The system shall create a user interface for adjusting the camera view. *Verification method*: test.
[SW-REQ-DEPTH-3]	(Functional) The system shall generate depth map of the 3D model from a defined camera view. *Verification method*: analysis.

[SW-REQ-DEPTH-4]	(Functional) The system shall generate a segmentation mask of the 3D model from a defined camera view. *Verification method*: analysis.
[SW-REQ-DEPTH-5]	(Functional) The system shall generate a scaleable gray-scale depth image of the 3D model from a defined camera view. *Verification method*: test.
[SW-REQ-DEPTH-6]	(Functional) The system shall write to file the generated files at the end of execution. *Verification method*: test.

4.5. Design Solution Definition

With the requirements defined and the trade studies performed, the design solution can be defined. It is chosen to use OctoMap due to its versatility and our requirement of designing derived octree classes with grasp quality attributes. Also, it is chosen to utilise pure C++ as software communication layer, due to the Software's limited distributed interaction which mainly falls back on text/binary files (see Figure 4.5). Below are the design solution definitions of the project's software subsystems.

4.5.1. OctoMap Grasping

There is the need for a set of classes that are capable of storing the properties relevant to the grasp analysis, as well as providing convenient methods for achieving said goals.

The responsibilities of this package are:

- Store parametric (voxel) 3D model of targets and grippers;
- Allow the storage of abstract properties the voxels of the model;
- Provide methods for geometric analyses and visualisation;
- Import binary tree models outputted from the 3D model voxelisation pipeline;
- Convert parametric 3D model stored in class to visualisable colourised model, using the abstract properties of the voxels as variables for the colour selection.

Two derived classes shall be created, one for gripper objects, and one for target objects (`OcTreeGraspQuality`). The gripper derived class shall contain a flag tagging voxels that are within the graspable region of the gripper (where interaction with the target is desirable). The target derived class shall contain nodes embedded with the grasp quality score associated with a grasp attempt with *that* node as center of grasp.

4.5.2. Grasp Planning Engine

The grasp planning engine is responsible for the control and processing of the derived octree types. It shall provide methods for interacting with the data in full (global analysis), as well as for partial state-space traversals for debugging and data processing. The grasp candidate sampling method used is semi-optimised instantiation process that locks the β and γ orientation of the grasp, such that the gripper always approaches the target surface along the surface's normal vector. Then, grasps are sampled at 4 roll orientations for every voxel in the target surface.

The most fundamental aspects of the grasp planning engine that drive performance are the grasp quality methods definitions, which are tabulated in Table 4.3. All the methods proposed shall rely on geometric information for grasp quality processing, which will effectively be abstractions and implementations of the following two fundamental model-interaction processing methods:

1. Align the two octrees (target and gripper) by transposing the target tree origin (see Figure 4.8) to coincide with the target node coordinates. Then iterate over the trees and consume the information regarding collisions of voxels in both trees to determine grasp quality.

2. Align the two trees (target and gripper) as before, but then cast rays from every voxel adjacent to the anti-podal plates of the gripper tree towards the symmetric voxel at the opposite antipodal plate. When and if a ray hits a target voxel, record its properties and use its position for contact surface analysis. Figure 4.7 provides a visual representation of rays being cast from both antipodal planes, which stop at the first target voxel that they hit.

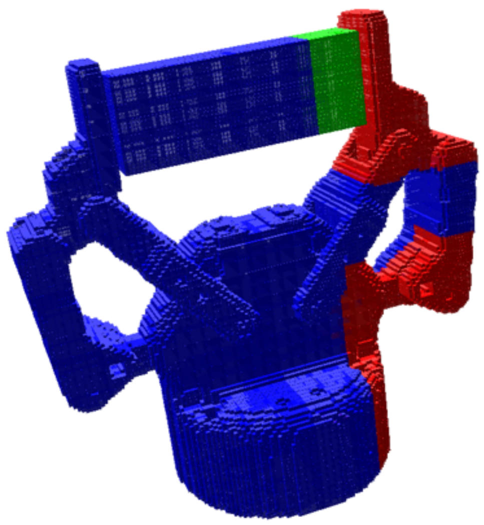


Figure 4.6: Gripper object displaying positively-interacting target voxels in solid green, and negatively-interacting target voxels in red.

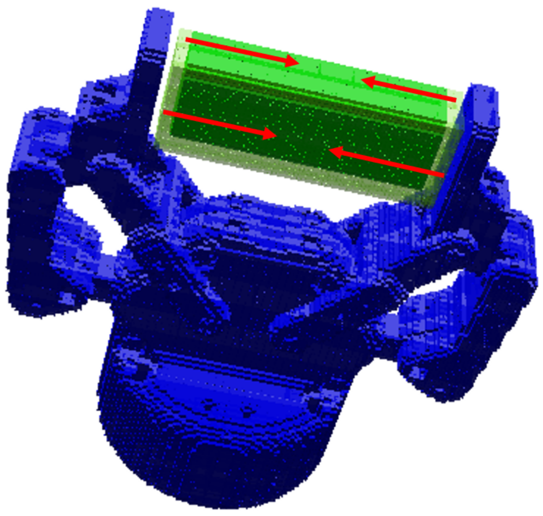


Figure 4.7: Gripper object with rays cast from its antipodal plates, as done in methods #2, #4, and #5.

4.5.3. 3D Model Voxelisation

The 3D model voxeliser shall provide the ability of inputting 3D models, such as Wavefronts, and obtaining as output voxelised representations of the model with customisable resolution. Given the availability of free tools to accomplish this goal, this functionality shall be a manual pipeline consisting of off-the-shelf tools that ultimately accomplish said objective.

Table 4.3: Overview of grasp quality methods proposed for the Aut-O-MAGIC project.

ID	Method	Description	Parameters
#1	Voxel superimposition	Count the number of target voxels within the graspable region of the gripper, and subtract the number of voxels that collide against the body of the gripper	Reward, Penalty
#2	Surface normals histogram	Build histogram of alignment angles between gripper antipodal plates normal direction and the target surface normal that they touch. Score is a combination of standard deviation of histogram and mean angle between the two vectors. Collisions of target body against the gripper apply a light penalty to the score	Mean/std_dev fraction, std_dev saturation
#3	Voxel superimposition + surface normals histogram	Methods #1 and #2 combined, each with 50% weight	#1 + #2
#4	Surface normals with discrete scoring	Cast rays in both directions between both antipodal gripper planes and note the first target node (in between the gripper planes) that gets hit by the ray. Similar to Method #2 but with discrete (individual scores), which cross the 0 graspability score at a defined angle value, instead of histogram generation. Collisions of target body against the gripper apply a light penalty to the score	Zero-score crossing
#5	Coplanarity contact points	Compute coplanarity of (separate) target surface contact points hit by rays cast from both antipodal gripper planes, using the disparity between the median and mean as well as the standard deviation of the height difference. Collisions of target body against the gripper apply a light penalty to the score	disparity/std_dev fraction, std_dev saturation, discretisation steps
#6	Voxel superimposition + surface normals with discrete scoring	Methods #1 and #4 combined, each with 50% weight	#1 + #4
#7	Voxel superimposition + coplanarity contact points	Methods #1 and #5 combined, each with 50% weight	#1 + #5

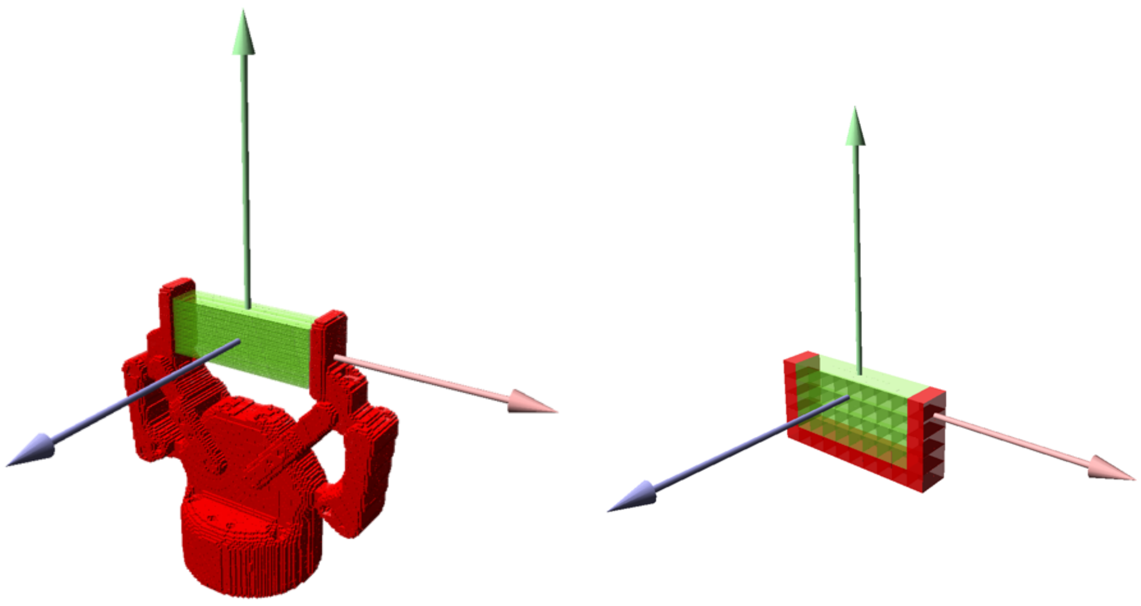


Figure 4.8: Origin of coordinates of a gripper tree in *loaded* state is at the center of the graspable region.

4.5.4. 3D Model to Depth Map

The goal of this utility is to provide a simple way to generate depth maps (Z-Buffers) from 3D models, to be used as synthetic inputs for neural networks and other relevant tasks. This is required to enable comparison of this thesis work with other works that use depth images as input. It provides:

- Depth map in [m] as a 3D `npy` file;
- Depth map in `png` format with a user-defined scaling factor (default = 1000.0), so that 255RGB corresponds to 255[mm] depth;
- Segmentation mask in `png` format;
- Intrinsic camera parameters in format used in the CQCNN package by Berkeley Automation.

4.5.5. Scene Reconstruction

The scene reconstruction from sensor data is decomposed into three blocks:

- Sensor driver that outputs point clouds captured by depth sensor at each time step;
- Point cloud scene reconstruction engine that aligns the clouds and merges them into a singular global point cloud;
- Point cloud to binary tree converter.

The scene reconstruction chain is decomposed into discrete steps to decrease development time by using the application most fit for each purpose. The sensor driver is implemented as a ROS2 node, the point cloud scene reconstruction engine in MATLAB, and the point cloud to binary tree converter in pure C++. Furthermore, the point cloud scene reconstruction engine is composed of the following processes:

1. K-NN noise filtering;
2. Uniform-distribution downsampling for faster and more accurate ICP point cloud pair-wise registration [68];
3. Odometry to calculate the rough motion from point to point, using ICP Point-to-Point pair-wise point cloud registration;
4. Loop closure to detect identical features in non-consecutive snapshots and create a connection between them for better transformation estimation than odometry alone. Performed using MATLAB context descriptors for fast filtering of bad candidates and then, on good candidates, the afore ICP Point-to-Point pair-wise point cloud registration algorithm is applied again;
5. Final global point cloud alignment of the de-noised, full-size, point clouds.

4.6. Conclusions

This chapter described the details of the system design for this project. It began with the technical requirements definition, followed by trade studies that led to the system hardware selection and software decomposition which, in turn, dictated the low-level derived technical requirements. The system is composed of the Microsoft K4A RGB-D camera, and four software modules. These modules are divided to comply with the single-responsibility principle, and together are able to:

1. Integrate sensor data to compile a voxel tree of the target object;
2. Voxelise a synthetic 3D model of a target object;
3. Perform global graspability analyses of a target object via its voxel tree, using a user-configurable gripper voxel model;
4. Interface with other state-of-the-art grasp planning solutions that rely on depth images for grasp planning, instead of 3D maps.

5 System Implementation

In this chapter the implementation details of the design subjects defined in section 4.5 are elaborated upon. For each subject, Unified Modelling Language (UML) Class Diagrams and dependency graphs are provided (if applicable), as well as other details of the design implementations, such as the standards used for the gripper models and file types used for I/O. The operation of software is documented in Appendix A to separate academic design relevant to this thesis work from purely technical system operation. Because the 3D model voxeliser pipeline is solely composed of external tools, its usage is documented in Appendix A and the software pipeline is omitted from this chapter. Similarly, the scene reconstruction engine is composed of the blocks already explicitly described in subsection 4.5.5, and implemented using off-the-shelf 3D processing functions from MATLAB, so it is also omitted from this chapter.

5.1. OctoMap Grasping

The OctoMap derived classes[1] are the OcTreeGraspQuality class, which is intended for handling a target object (i.e. the satellite to grasp), and the OcTreeGripper class, intended for the gripper object. They extend the generic functionality provided by OctoMap's OcTree and OcTreeNode (voxel object) by enabling:

- Store grasp quality at discretised gripper rotation angles in each voxel;
- Convert objects to colourised octree for visualisation purposes;
- Draw a Bounding Box (BBX) between the gripper antipodal grasping plates to label them are region where target voxels *is* desirable;
- Custom single-solution surface normal determination function;
- Import of binary voxel trees into object.

The UML Class Diagram for this package is provided in Figure 5.1, and the dependency graphs for the two classes developed in Figure 5.2 and Figure 5.3 for the OcTreeGraspQuality class and the OcTreeGripper class, respectively.

[1] `https://github.com/aut-o-magic/octomap-grasping`

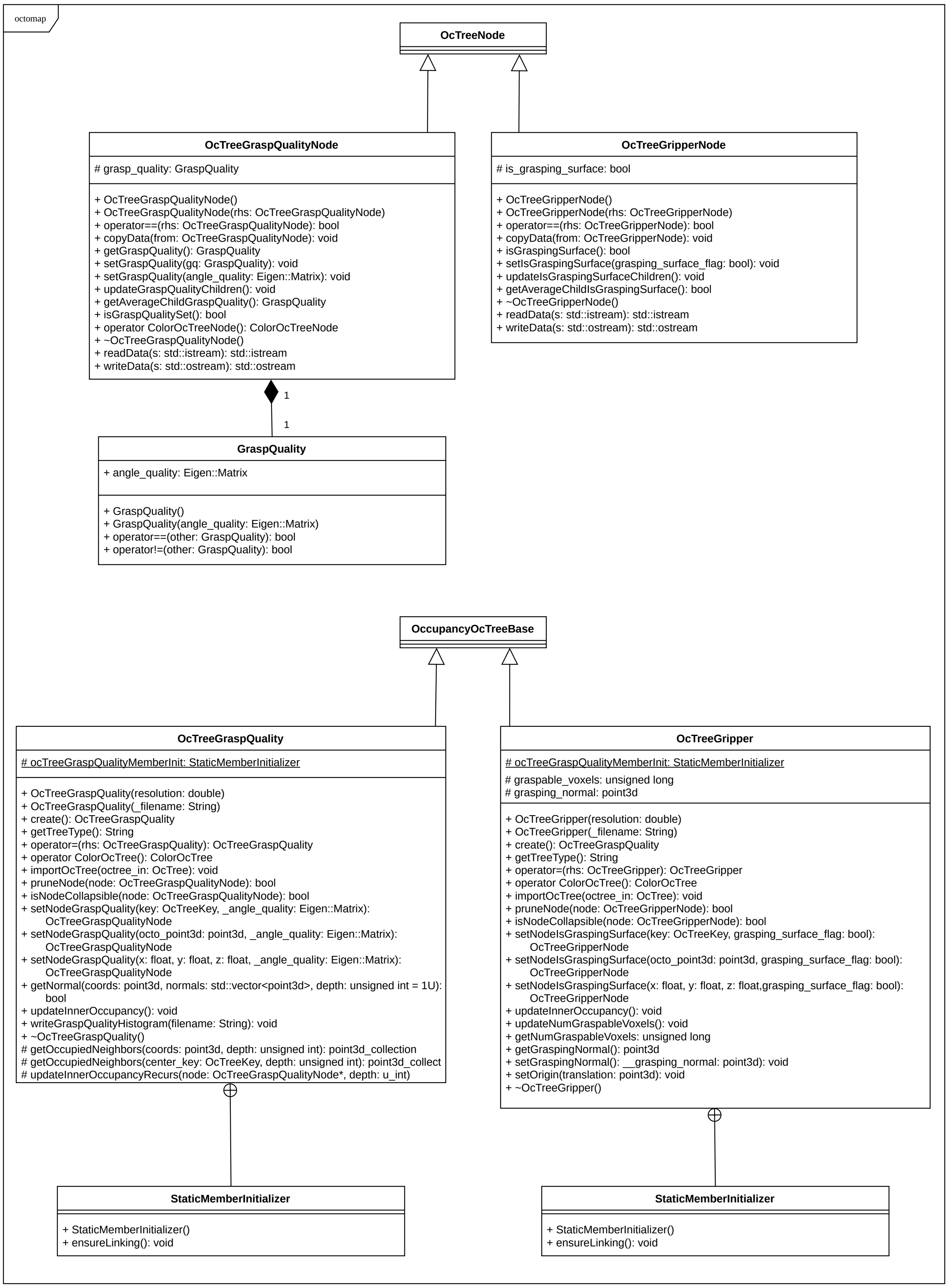

octomap
OcTreeNode
OcTreeGraspQualityNode
# grasp_quality: GraspQuality
+ OcTreeGraspQualityNode()
+ OcTreeGraspQualityNode(rhs: OcTreeGraspQualityNode)
+ operator==(rhs: OcTreeGraspQualityNode): bool
+ copyData(from: OcTreeGraspQualityNode): void
+ getGraspQuality(): GraspQuality
+ setGraspQuality(gq: GraspQuality): void
+ setGraspQuality(angle_quality: Eigen::Matrix): void
+ updateGraspQualityChildren(): void
+ getAverageChildGraspQuality(): GraspQuality
+ isGraspQualitySet(): bool
+ operator ColorOcTreeNode(): ColorOcTreeNode
+ ~OcTreeGraspQualityNode()
+ readData(s: std::istream): std::istream
+ writeData(s: std::ostream): std::ostream
OcTreeGripperNode
# is_grasping_surface: bool
+ OcTreeGripperNode()
+ OcTreeGripperNode(rhs: OcTreeGripperNode)
+ operator==(rhs: OcTreeGripperNode): bool
+ copyData(from: OcTreeGripperNode): void
+ isGraspingSurface(): bool
+ setIsGraspingSurface(grasping_surface_flag: bool): void
+ updateIsGraspingSurfaceChildren(): void
+ getAverageChildIsGraspingSurface(): bool
+ ~OcTreeGripperNode()
+ readData(s: std::istream): std::istream
+ writeData(s: std::ostream): std::ostream
1
1
GraspQuality
+ angle_quality: Eigen::Matrix
+ GraspQuality()
+ GraspQuality(angle_quality: Eigen::Matrix)
+ operator==(other: GraspQuality): bool
+ operator!=(other: GraspQuality): bool
OccupancyOcTreeBase
OcTreeGraspQuality
# ocTreeGraspQualityMemberInit: StaticMemberInitializer
+ OcTreeGraspQuality(resolution: double)
+ OcTreeGraspQuality(_filename: String)
+ create(): OcTreeGraspQuality
+ getTreeType(): String
+ operator=(rhs: OcTreeGraspQuality): OcTreeGraspQuality
+ operator ColorOcTree(): ColorOcTree
+ importOcTree(octree_in: OcTree): void
+ pruneNode(node: OcTreeGraspQualityNode): bool
+ isNodeCollapsible(node: OcTreeGraspQualityNode): bool
+ setNodeGraspQuality(key: OcTreeKey, _angle_quality: Eigen::Matrix): OcTreeGraspQualityNode
+ setNodeGraspQuality(octo_point3d: point3d, _angle_quality: Eigen::Matrix): OcTreeGraspQualityNode
+ setNodeGraspQuality(x: float, y: float, z: float, _angle_quality: Eigen::Matrix): OcTreeGraspQualityNode
+ getNormal(coords: point3d, normals: std::vector<point3d>, depth: unsigned int = 1U): bool
+ updateInnerOccupancy(): void
+ writeGraspQualityHistogram(filename: String): void
+ ~OcTreeGraspQuality()
# getOccupiedNeighbors(coords: point3d, depth: unsigned int): point3d_collection
# getOccupiedNeighbors(center_key: OcTreeKey, depth: unsigned int): point3d_collect
# updateInnerOccupancyRecurs(node: OcTreeGraspQualityNode*, depth: u_int)
OcTreeGripper
# ocTreeGraspQualityMemberInit: StaticMemberInitializer
# graspable_voxels: unsigned long
# grasping_normal: point3d
+ OcTreeGripper(resolution: double)
+ OcTreeGripper(_filename: String)
+ create(): OcTreeGraspQuality
+ getTreeType(): String
+ operator=(rhs: OcTreeGripper): OcTreeGripper
+ operator ColorOcTree(): ColorOcTree
+ importOcTree(octree_in: OcTree): void
+ pruneNode(node: OcTreeGripperNode): bool
+ isNodeCollapsible(node: OcTreeGripperNode): bool
+ setNodeIsGraspingSurface(key: OcTreeKey, grasping_surface_flag: bool): OcTreeGripperNode
+ setNodeIsGraspingSurface(octo_point3d: point3d, grasping_surface_flag: bool): OcTreeGripperNode
+ setNodeIsGraspingSurface(x: float, y: float, z: float,grasping_surface_flag: bool): OcTreeGripperNode
+ updateInnerOccupancy(): void
+ updateNumGraspableVoxels(): void
+ getNumGraspableVoxels: unsigned long
+ getGraspingNormal(): point3d
+ setGraspingNormal(): __grasping_normal: point3d): void
+ setOrigin(translation: point3d): void
+ ~OcTreeGripper()
StaticMemberInitializer
+ StaticMemberInitializer()
+ ensureLinking(): void
StaticMemberInitializer
+ StaticMemberInitializer()
+ ensureLinking(): void

Figure 5.1: UML Class Diagram for Octomap-Grasping package.

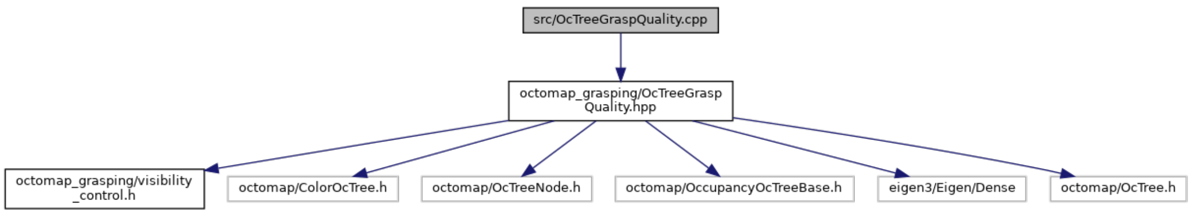

src/OcTreeGraspQuality.cpp
octomap_grasping/OcTreeGraspQuality.hpp
octomap_grasping/visibility_control.h
octomap/ColorOcTree.h
octomap/OcTreeNode.h
octomap/OccupancyOcTreeBase.h
eigen3/Eigen/Dense
octomap/OcTree.h

Figure 5.2: Dependency graph for `OcTreeGraspQuality.cpp` source file.

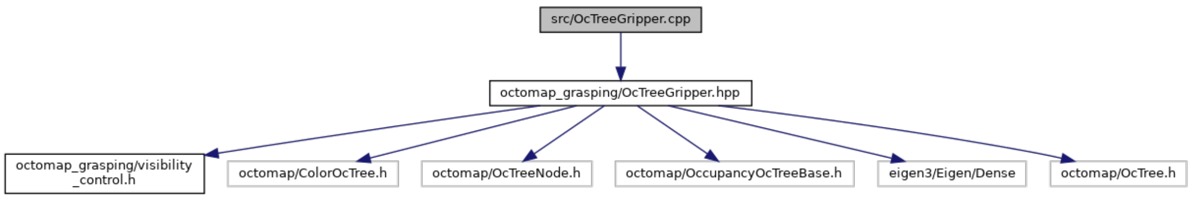

src/OcTreeGripper.cpp
octomap_grasping/OcTreeGripper.hpp
octomap_grasping/visibility_control.h
octomap/ColorOcTree.h
octomap/OcTreeNode.h
octomap/OccupancyOcTreeBase.h
eigen3/Eigen/Dense
octomap/OcTree.h

Figure 5.3: Dependency graph for `OcTreeGripper.cpp` source file.

5.2. Grasp Planning Engine

The Grasp Planning Engine[2] enables grasp planning operations using the OctoMap Grasping classes defined in section 5.1. It provides the necessary methods for meeting the system architecture model defined in Figure 4.5, as well as having convenience functions for performing individual ("local") grasp quality analyses instead of only global ones. In Figure 5.4, the UML Class Diagram for the grasp planning engine package is provided, while in Figure 5.5 the dependency graph of the grasp planning engine entry point (`gp_node.cpp`) and the rest of interacting packages is depicted. For grasp analyses, the Robotiq 2F85 gripper is used. In Figure 5.6, the gripper is depicted with the graspable region ($8.4 \times 3.6 \times 1.9$cm) drawn in semi-transparent green, while the body of the gripper is drawn in red. Because of the significant size of this gripper model, however, a simplified abstraction of it is used for the global graspability map. This simplified gripper is depicted in Figure 5.7 and has the identical grasping region as its larger counterpart. The only difference between the two grippers is the amount of body voxels attached to the model. In preliminary tests it was observed how, due to the fact that the gripper approaches the target model following the surface normal direction, the gripper body voxels below the graspable region rarely interact with the target body. Eliminating those allows for much faster processing at a minimal expense.

[2]`https://github.com/aut-o-magic/grasp-planning-engine`

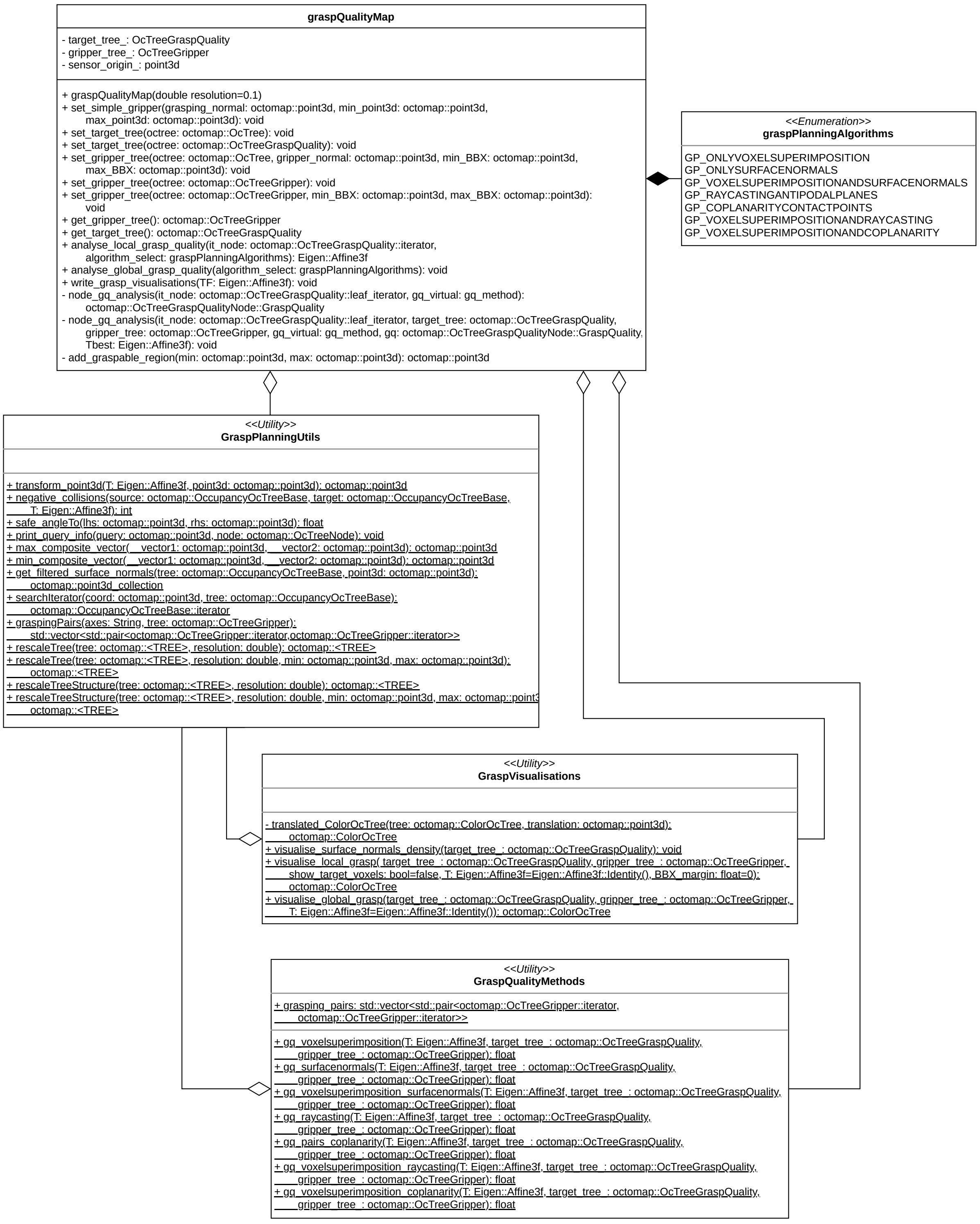

graspQualityMap
- target_tree_: OcTreeGraspQuality
- gripper_tree_: OcTreeGripper
- sensor_origin_: point3d
+ graspQualityMap(double resolution=0.1)
+ set_simple_gripper(grasping_normal: octomap::point3d, min_point3d: octomap::point3d, max_point3d: octomap::point3d): void
+ set_target_tree(octree: octomap::OcTree): void
+ set_target_tree(octree: octomap::OcTreeGraspQuality): void
+ set_gripper_tree(octree: octomap::OcTree, gripper_normal: octomap::point3d, min_BBX: octomap::point3d, max_BBX: octomap::point3d): void
+ set_gripper_tree(octree: octomap::OcTreeGripper): void
+ set_gripper_tree(octree: octomap::OcTreeGripper, min_BBX: octomap::point3d, max_BBX: octomap::point3d): void
+ get_gripper_tree(): octomap::OcTreeGripper
+ get_target_tree(): octomap::OcTreeGraspQuality
+ analyse_local_grasp_quality(it_node: octomap::OcTreeGraspQuality::iterator, algorithm_select: graspPlanningAlgorithms): Eigen::Affine3f
+ analyse_global_grasp_quality(algorithm_select: graspPlanningAlgorithms): void
+ write_grasp_visualisations(TF: Eigen::Affine3f): void
- node_gq_analysis(it_node: octomap::OcTreeGraspQuality::leaf_iterator, gq_virtual: gq_method): octomap::OcTreeGraspQualityNode::GraspQuality
- node_gq_analysis(it_node: octomap::OcTreeGraspQuality::leaf_iterator, target_tree: octomap::OcTreeGraspQuality, gripper_tree: octomap::OcTreeGripper, gq_virtual: gq_method, gq: octomap::OcTreeGraspQualityNode::GraspQuality, Tbest: Eigen::Affine3f): void
- add_graspable_region(min: octomap::point3d, max: octomap::point3d): octomap::point3d
<<Enumeration>>
graspPlanningAlgorithms
GP_ONLYVOXELSUPERIMPOSITION
GP_ONLYSURFACENORMALS
GP_VOXELSUPERIMPOSITIONANDSURFACENORMALS
GP_RAYCASTINGANTIPODALPLANES
GP_COPLANARITYCONTACTPOINTS
GP_VOXELSUPERIMPOSITIONANDRAYCASTING
GP_VOXELSUPERIMPOSITIONANDCOPLANARITY
<<Utility>>
GraspPlanningUtils
+ transform_point3d(T: Eigen::Affine3f, point3d: octomap::point3d): octomap::point3d
+ negative_collisions(source: octomap::OccupancyOcTreeBase, target: octomap::OccupancyOcTreeBase, T: Eigen::Affine3f): int
+ safe_angleTo(lhs: octomap::point3d, rhs: octomap::point3d): float
+ print_query_info(query: octomap::point3d, node: octomap::OcTreeNode): void
+ max_composite_vector(__vector1: octomap::point3d, __vector2: octomap::point3d): octomap::point3d
+ min_composite_vector(__vector1: octomap::point3d, __vector2: octomap::point3d): octomap::point3d
+ get_filtered_surface_normals(tree: octomap::OccupancyOcTreeBase, point3d: octomap::point3d): octomap::point3d_collection
+ searchIterator(coord: octomap::point3d, tree: octomap::OccupancyOcTreeBase): octomap::OccupancyOcTreeBase::iterator
+ graspingPairs(axes: String, tree: octomap::OcTreeGripper): std::vector<std::pair<octomap::OcTreeGripper::iterator,octomap::OcTreeGripper::iterator>>
+ rescaleTree(tree: octomap::<TREE>, resolution: double): octomap::<TREE>
+ rescaleTree(tree: octomap::<TREE>, resolution: double, min: octomap::point3d, max: octomap::point3d): octomap::<TREE>
+ rescaleTreeStructure(tree: octomap::<TREE>, resolution: double): octomap::<TREE>
+ rescaleTreeStructure(tree: octomap::<TREE>, resolution: double, min: octomap::point3d, max: octomap::point3
octomap::<TREE>
<<Utility>>
GraspVisualisations
- translated_ColorOcTree(tree: octomap::ColorOcTree, translation: octomap::point3d): octomap::ColorOcTree
+ visualise_surface_normals_density(target_tree_: octomap::OcTreeGraspQuality): void
+ visualise_local_grasp(target_tree_: octomap::OcTreeGraspQuality, gripper_tree_: octomap::OcTreeGripper, show_target_voxels: bool=false, T: Eigen::Affine3f=Eigen::Affine3f::Identity(), BBX_margin: float=0): octomap::ColorOcTree
+ visualise_global_grasp(target_tree_: octomap::OcTreeGraspQuality, gripper_tree_: octomap::OcTreeGripper, T: Eigen::Affine3f=Eigen::Affine3f::Identity()): octomap::ColorOcTree
<<Utility>>
GraspQualityMethods
+ grasping_pairs: std::vector<std::pair<octomap::OcTreeGripper::iterator, octomap::OcTreeGripper::iterator>>
+ gq_voxelsuperimposition(T: Eigen::Affine3f, target_tree_: octomap::OcTreeGraspQuality, gripper_tree_: octomap::OcTreeGripper): float
+ gq_surfacenormals(T: Eigen::Affine3f, target_tree_: octomap::OcTreeGraspQuality, gripper_tree_: octomap::OcTreeGripper): float
+ gq_voxelsuperimposition_surfacenormals(T: Eigen::Affine3f, target_tree_: octomap::OcTreeGraspQuality, gripper_tree_: octomap::OcTreeGripper): float
+ gq_raycasting(T: Eigen::Affine3f, target_tree_: octomap::OcTreeGraspQuality, gripper_tree_: octomap::OcTreeGripper): float
+ gq_pairs_coplanarity(T: Eigen::Affine3f, target_tree_: octomap::OcTreeGraspQuality, gripper_tree_: octomap::OcTreeGripper): float
+ gq_voxelsuperimposition_raycasting(T: Eigen::Affine3f, target_tree_: octomap::OcTreeGraspQuality, gripper_tree_: octomap::OcTreeGripper): float
+ gq_voxelsuperimposition_coplanarity(T: Eigen::Affine3f, target_tree_: octomap::OcTreeGraspQuality, gripper_tree_: octomap::OcTreeGripper): float

Figure 5.4: UML Class Diagram for Grasp Planning Engine package.

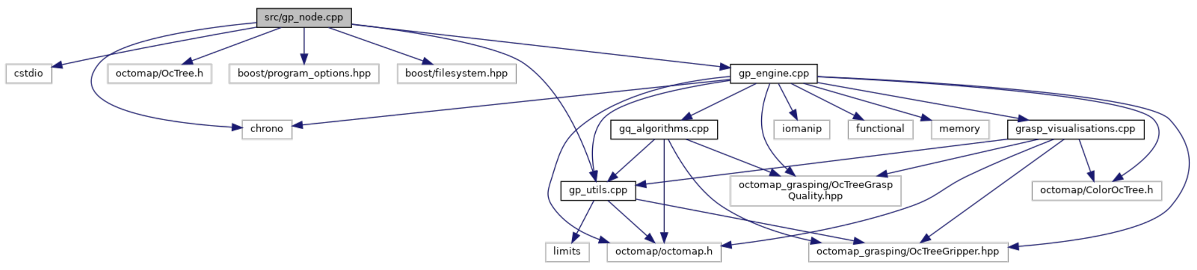


Figure 5.5: Dependency graph for `gp_node.cpp` source file.

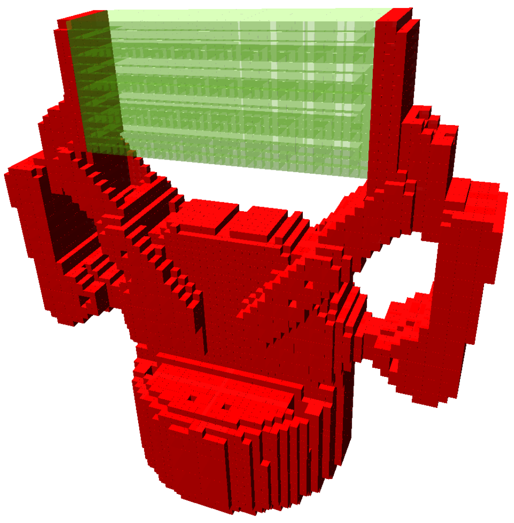


Figure 5.6: Robotiq 2F85 gripper with graspable region between the antipodal grasping plates.

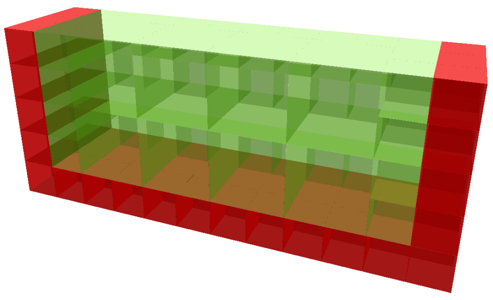


Figure 5.7: Simple gripper with graspable region between the antipodal grasping plates.

5.3. 3D Model to Depth Map

The 3D Model to 2D depth image projection tool[3] is used to benchmark the grasp planning of this thesis work with other state-of-the-art solutions. There are numerous standards used to express depth data, so the code of this utility is designed to be modular and easy to edit for custom implementations.

The utility is written as a procedural program in Python 3.8 and uses Open3D as a backend for mesh processing [69], and PyPNG for image I/O operations. Given that the utility was not designed following an Object-Oriented Design (OOD) approach because of its simplicity, no UML Class diagrams are provided in this section. As a result, the utility generates the items described below. An example processing of the 3D model depicted in Figure 5.8 is performed to visualise the operation.

- Depth map in [m] as a 3D `npy` file;
- Depth map in `png` format (depicted in Figure 5.9). By default it scales the metric scale up by `1000`, meaning that a depth of 200mm would depict with a colour intensity of 200 (out of 255);
- Segmentation mask in `png` format (Figure 5.10);
- Intrinsic camera parameters in format used in the CQCNN package by Berkeley Automation listed in Script 5.1.

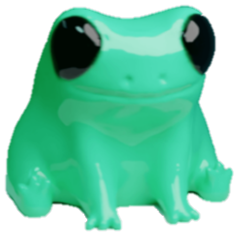


Figure 5.8: Small frog 3D model used to generate sample outputs.

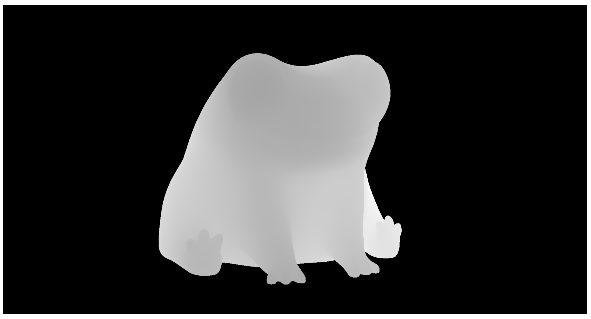


Figure 5.9: Example depth map output.

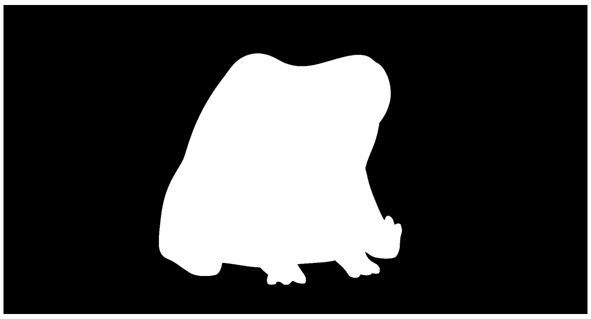


Figure 5.10: Example segmentation mask output.

Script 5.1: Contents of intrinsic camera parameters output file `virtualcam.intr`

```
{"_cy": 499.500000, "_cx": 499.500000, "_fy": 3000.000000, "_height": 1000, "_fx": 3000.000000, "_width": 1000, "_skew": 0.000000, "_K": 0, "_frame": "virtualcam"}
```

[3]https://github.com/aut-o-magic/3DtoDepthmap

III

Framework Evaluation

6 System Verification

In this chapter, qualitative analyses of the performance of the grasp quality methods (and the overall grasp planning engine) developed in the scope of this work are executed. Due to the geometric nature of the processing, results will contain numerous illustrations as they are ultimately the most native way to judge and analyse geometric characteristics.

This chapter is divided in two analysis sequences:

- Envisat satellite grasp quality analysis with qualitative grasp planning method comparison. The goal is to find the better performing methods.
- Dex-Net comparison, where grasp candidates of this thesis work are compared against the largely-consensuated state-of-the-art in empirical grasp planning.

Lastly, conclusions are drawn on the performance of the system on the synthetic verification data.

6.1. Envisat

A qualitative assessment of the graspability of Envisat is performed with each of the grasp planning methods developed. Envisat is a large passive satellite still in-orbit, which was used for Earth observation. It has been a candidate for active debris removal in the past, due to its very slow orbit decay and chance of collision with other debris [70]. This led to it being a well-documented satellite for academic work.

The results of the graspability assessment of Envisat, tabulated in Table 6.1 and depicted below, show that the properly tuned metrics used for grasp planning are sensible and suggest reasonable grasp candidates.

The analyses are encoded based on the codebase version number and the grasp quality method used. Then, a mark "Mk" is added when the parameters of the grasp quality method are altered. For example, `R61 Mk1` corresponds to codebase version 6, grasp planning method 1 (voxel superimposition), mark 1. Although not all of the tabulated analyses are from codebase version 8, they are fully comparable because the codebase between version 6 and 8 did not change for analyses made with grasp planning method 1.

Table 6.1: Parameter matrix of the parameters used for the qualitative analysis of Envisat.

Analysis ID	Reward	Penalty	Angle saturation (°)	Zero Crossing (°)	Bin saturation	Bin steps
R61 Mk0	1	1				
R61 Mk1	1	2				
R61 Mk2	1	5				
R61 Mk3	1	50				
R82 Mk0			1			
R82 Mk1			10			
R83 Mk0	1	1	1			
R83 Mk1	1	10	10			
R84 Mk0				45		
R84 Mk1				30		
R85 Mk0					5	100
R85 Mk1					50	100
R86 Mk0	1	1		45		
R86 Mk1	1	10		30		
R87 Mk0	1	1			5	100
R87 Mk1	1	10			50	100

From these analyses, it is observed how, with the right parameter tuning, all grasp planning methods considered except for method #6 appear to highlight regions that appear reasonable to grasp. Nevertheless, methods #1, #5, and #7 are the ones that show particularly good results. In the visualisations of these results below, it can be easily observed how the grasping regions around thin edges are strongly favoured, while planar and concave regions floored to negligible grasp quality. Major differences between the pure voxel superimposition method #1 and the composite methods #5 and #7 is that the former also strongly rejects thin connectors, while the other two accept grasp candidates on those regions to varying degrees. In essence, method #1 directly measures grasp quality, while method #5 aids in the robustness of the solution by rewarding highly planar grasping surfaces. Method #7 then, as defined in previous sections, is a weighted average of #1 and #5. Visualisations for the rejected methods can be found in Appendix B.

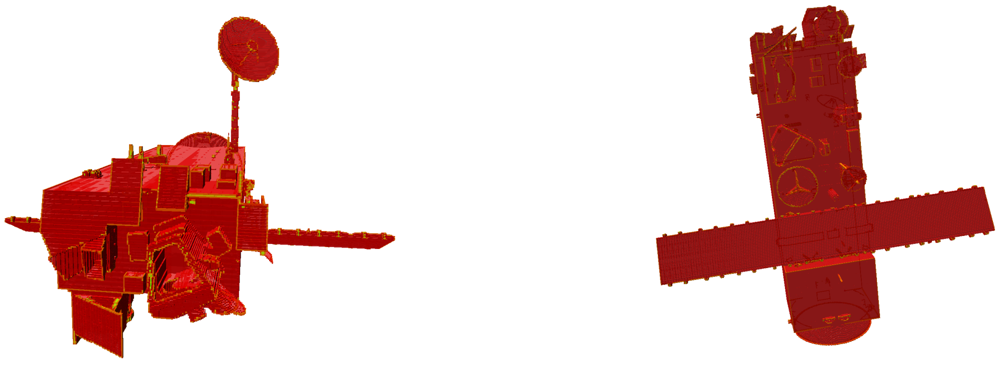


Figure 6.1: Envisat graspability as analysed by voxel superimposition method (#1) with Reward = 1, and Penalty = 50.

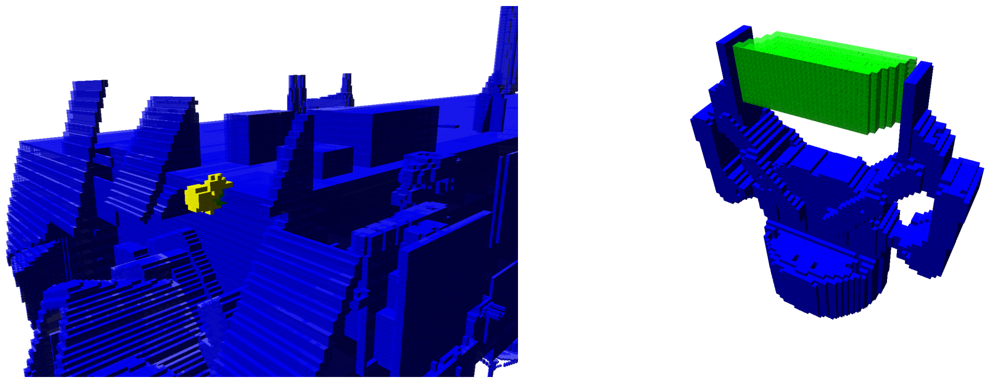


Figure 6.2: Local view of best grasp candidate for method #1 analysis. Grasp quality = 0.833333.

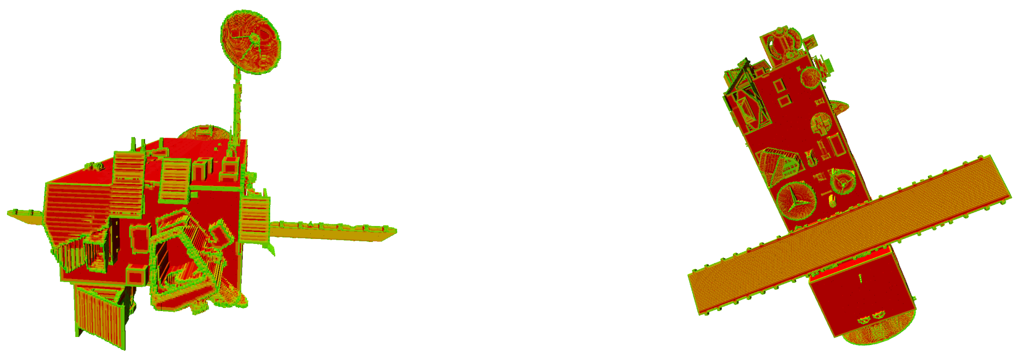


Figure 6.3: Envisat graspability as analysed by coplanarity of contact points method (#5) with Bin saturation = 50 and Bin steps = 100

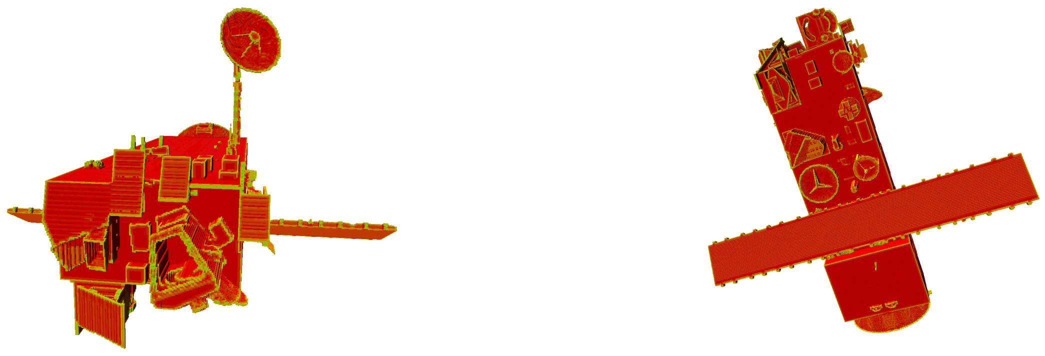


Figure 6.4: Envisat graspability as analysed by voxel superimposition + coplanarity of contact points (#7) with Reward = 1, Penalty = 10, Bin saturation = 50, and Bin steps = 100.

6.2. Dex-Net Simulations

In this section, a comparison analysis is performed between the Aut-O-MAGIC solution and one of the best well-known deep learning grasp planning solutions, Dex-Net.

In the latest Dex-Net version for antipodal grasping, GQCNN-4.0-PJ, the camera used it is PhotoNeo Phoxi S [62]. The camera parameters are listed in Script 6.1 and inputted into the 3D model to depth map tool for depth image generation. The ABB YuMi Gripper used by UC Berkeley is custom designed, but with similar characteristics as to the default Robotiq 2F-85 gripper used in this thesis. Its maximum grasping width, however, is 5 cm[1], instead of 8cm for the Robotiq 2F85 gripper. Overall, their custom gripper is $5 \times 3 \times 2$ cm while ours is $8.3 \times 3.6 \times 1.9$ cm, making it somewhat wider but with negligible contact surface difference. Nevertheless, the gripper model in our grasp planning algorithm is adapted to match the UiMi custom gripper, by using the simple gripper model with adapted graspable region. The YuMi custom gripper model is depicted in Figure 6.5, while its simple gripper model used for benchmarking in Figure 6.6.

In general, it is good practise to keep in mind that deep learning models (and by extension the GQ-CNN model below) are sensitive to the parameters used during dataset generation, specifically, the GQ-CNN was trained under the following conditions:

- Gripper geometry: ABB YuMi Parallel Jaw Gripper;
- Camera intrinsics: PhotoNeo Phoxi S camera;
- Distance between camera and workspace during rendering: 50–70 cm for all the pre-trained models.

This conditions are accommodated for in this comparison by adjusting the gripper geometry to the custom ABB YuMi gripper used in Dex-Net, as well as using their camera model (simulating a PhotoNeo Phoxi S camera) to generate the depth images for analysis. Finally, the depth images generated are carefully crafted to be rendered at a distance 50–70cm from the virtual camera.

Script 6.1: Intrinsic camera parameters for the PhotoNeo Phoxi S. camera used to train the CQCNN-4.0-PJ model

```
{"_cy": 191.75, "_cx": 255.5, "_fy": 552.5, "_height": 386, "_fx":
552.5, "_width": 516, "_skew": 0.0, "_K": 0, "_frame": "phoxi"}
```

Comparative analyses between our solution and Dex-Net 4.0 are performed in two different objects, namely:

1. An adversarial pyramid object, depicted in Figure 6.7. This object has been generated for the purpose of these analyses. It attempts to force the grasp planner to decide within a *range of preference* whether it prefers grasping a thicker structure, or one with more parallel surface areas. The object is a cone that follows a non-linear slope between its narrower tip and its wider base. Its length is 0.5m, while the diameter ranges from 0.02m to 0.2m.

[1]https://github.com/BerkeleyAutomation/dex-net/blob/master/data/grippers/yumi_metal_spline/params.json

2. A complex (but easier to grasp) vase, depicted in Figure 6.15. This object was part of pool of 3D models used to train Dex-Net, so its performance on the Dex-Net grasp planning method should be sensible.

In the figures below, analyses of the adversarial pyramid using grasp planning method #1, #5, and #7 are provided. It can be seen how the global graspability maps clearly highlight a specific region of the pyramid, where the thickness is still fair and the slope is still reasonably low. The results from Dex-Net 4.0 are depicted in Figure 6.14 and Figure 6.16. The results of our grasp quality analyser for the vase object are represented following in a similar fashion to the Dex-Net labels to aid readability and provide an un-occluded view of the region. Surprisingly, for the vase object, the GQ-CNN deep learning model was unable to resolve any grasp candidates when the depth image was captured using the same orientation as observed in Figure 6.15. Only after rotating the model slightly it was able to identify a grasp candidate.

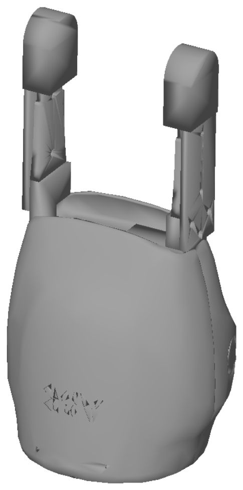


Figure 6.5: Custom ABB YuMi Gripper used by UC Berkeley for its GQCNN and Dex-Net projects. Graspable region (only area with padding) is approximately $5 \times 3 \times 2$ cm.

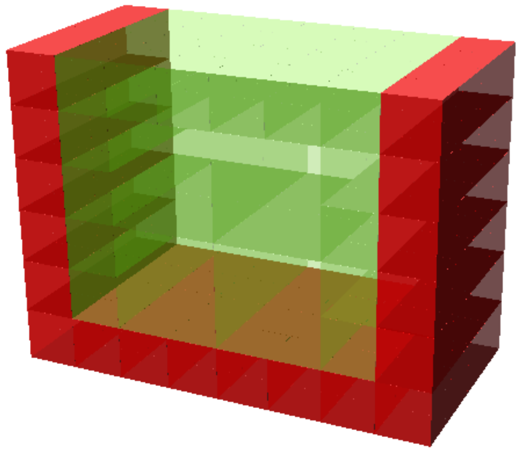


Figure 6.6: Representation of the custom ABB YuMi Gripper used by UC Berkeley for its GQCNN and Dex-Net projects. Graspable region is $5 \times 3 \times 2$ cm.

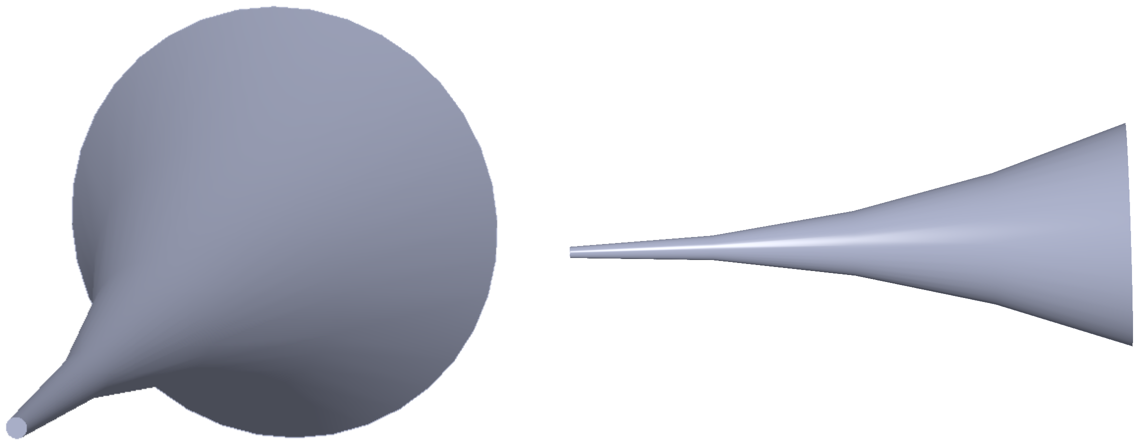


Figure 6.7: Overview of the adversarial pyramid object. Its length is 0.5 m, while the diameter ranges from 0.02 m to 0.2 m.

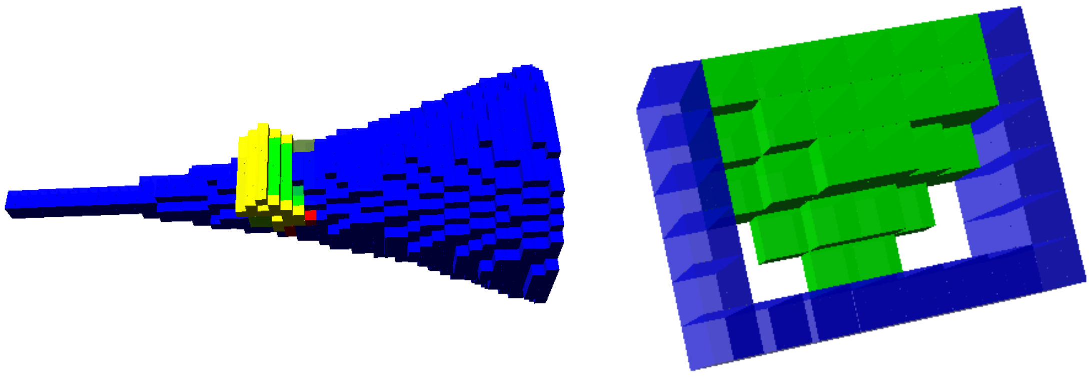


Figure 6.8: Best grasping candidate of the adversarial pyramid using the custom ABB YuMi gripper model and grasp planning algorithm #1. Right image is isolated view of target voxels (green) interacting with the gripper. Grasp quality score = 0.608333.

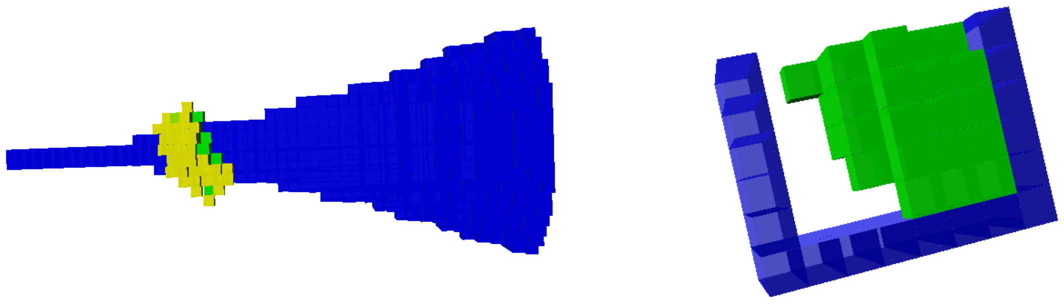


Figure 6.9: Best grasping candidate of the adversarial pyramid using the custom ABB YuMi gripper model and grasp planning algorithm #5. Right image is isolated view of target voxels (green) interacting with the gripper. Grasp quality score = 0.9.

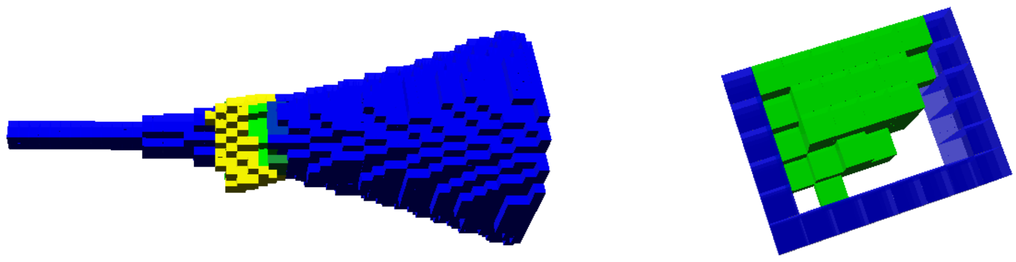


Figure 6.10: Best grasping candidate of the adversarial pyramid using the custom ABB YuMi gripper model and grasp planning algorithm #7. Right image is isolated view of target voxels (green) interacting with the gripper. Grasp quality score = 0.691667.

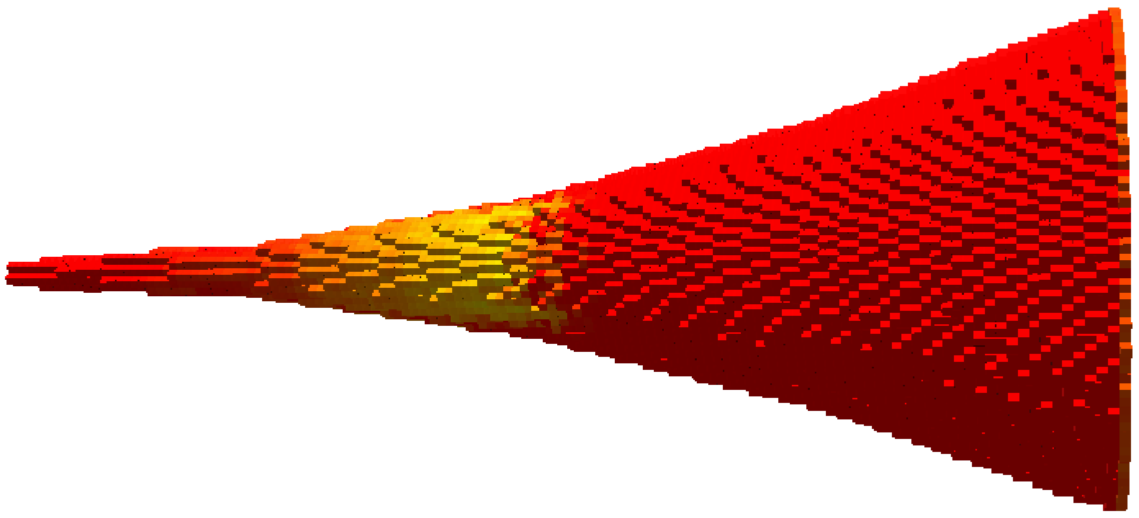


Figure 6.11: Global graspability map of the adversarial pyramid using the custom ABB YuMi gripper model and grasp planning algorithm #1.

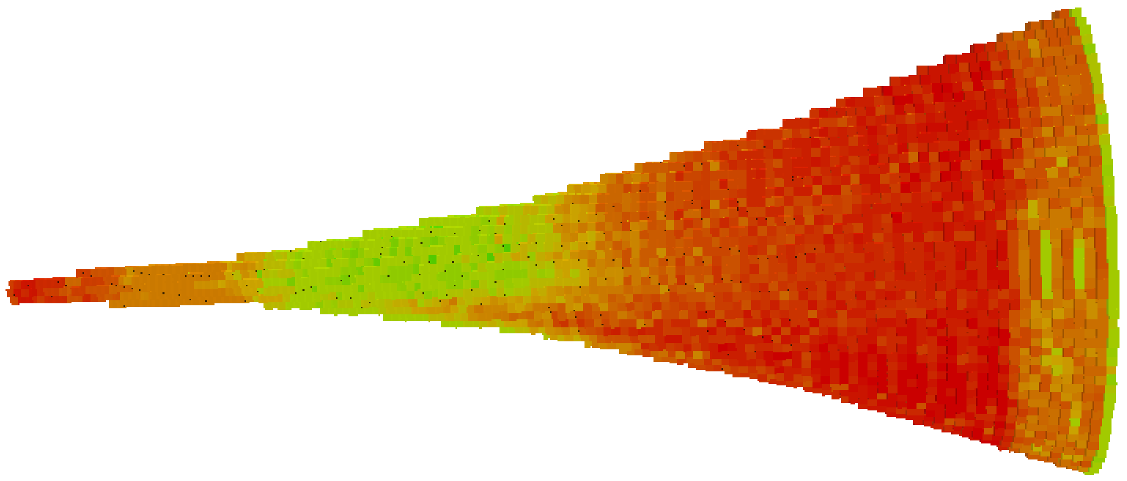


Figure 6.12: Global graspability map of the adversarial pyramid using the custom ABB YuMi gripper model and grasp planning algorithm #5.

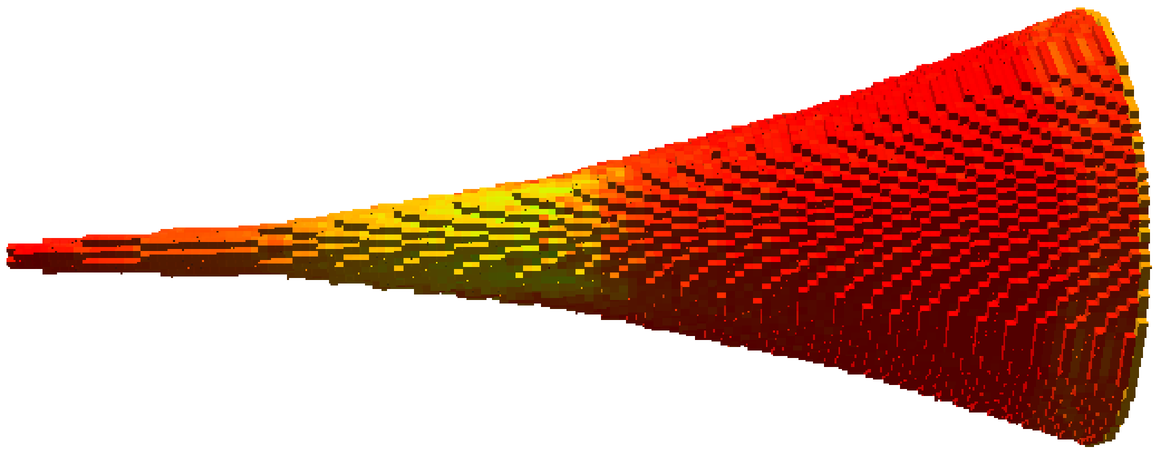


Figure 6.13: Global graspability map of the adversarial pyramid using the custom ABB YuMi gripper model and grasp planning algorithm #7.

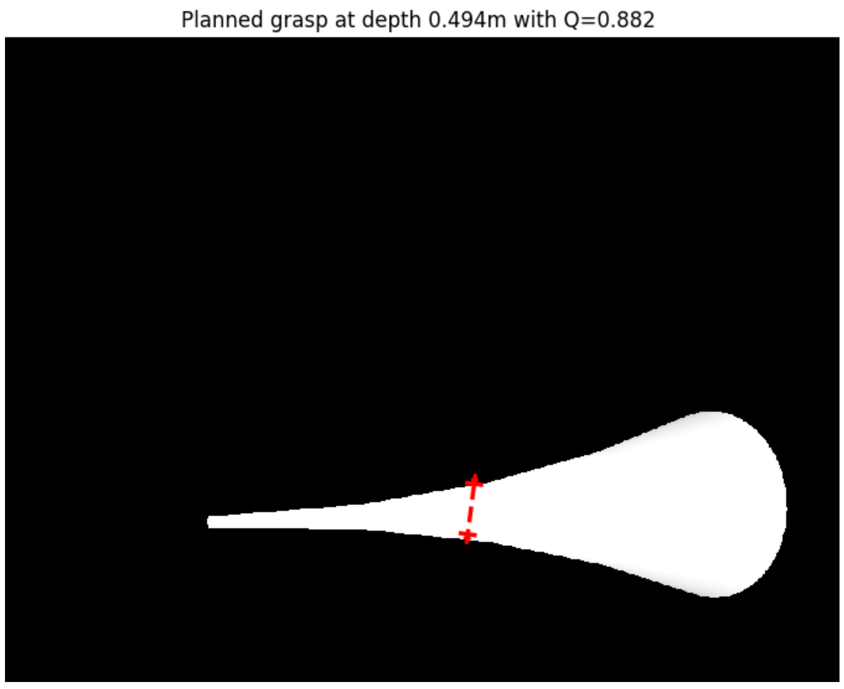


Figure 6.14: Best grasping candidate of the adversarial pyramid using the custom ABB YuMi gripper model ran on the Dex-Net 4.0 GQCNN-4.0-PJ deep learning model from UC Berkeley. Grasp quality score = 0.882.

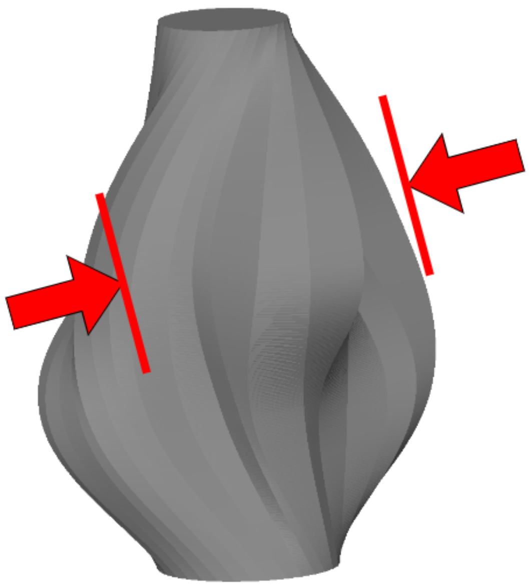


Figure 6.15: Illustrated best grasping candidate of the vase object using the custom ABB YuMi gripper model and grasp planning algorithm #7. Drawn antipodal plates to mimic GQCNN representation of the vase object for easier comparison. The region being grasped is a hill that corresponds to the hill on the right-hand side of Figure 6.16. Grasp quality score = 0.8.

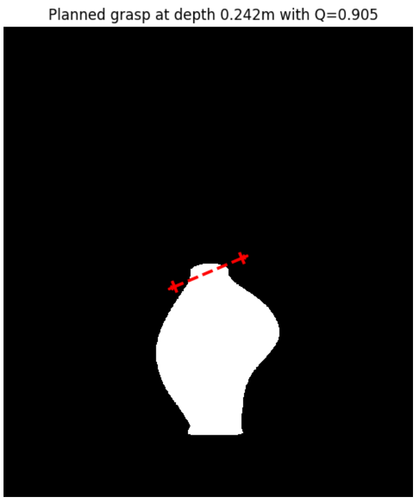


Figure 6.16: Best grasping candidate of the vase object using the custom ABB YuMi gripper model ran on the Dex-Net 4.0 GQCNN-4.0-PJ deep learning model from UC Berkeley. Grasp quality score = 0.905.

6.3. Conclusions

The global grasp quality analyses performed on synthetic data in this chapter show that the geometric grasp quality metrics proposed in this thesis work are valid. Out of the seven grasp quality methods proposed, it was demonstrated how the metrics including voxel superimposition scores between the target and gripper objects, as well as a measure of target surface contact points coplanarity, performed best. With iterative weight tuning, it was possible to achieve grasp quality candidates that are verifiable by inspection.

The results of the grasp quality analyses developed in this thesis were compared against results from the state-of-the-art grasp planning method Dex-Net 4, which uses deep learning. The comparisons showed agreement in regards to best grasp candidate selection, which verifies that the methods proposed in this work are sensible. Furthermore, comparison with the afore deep learning-based grasp planning method highlighted one of the shortcomings of deep learning methods that use image-based inputs, instead of 3D maps. It was observed how the grasp planning method developed in this thesis work was able to suggest better grasp candidates than the deep learning-based alternative for a particular adversarial object. This superiority is hypothesised to be due to the neural network being trained on small objects where grasps that are narrower than the object itself were not sufficiently present.

7 System Validation

To validate the resolution requirements of the system, its robustness to real-world noise, and its general applicability in real-world scenarios, a test campaign simulating a rendezvous scenario is performed in the GNC Rendezvous, Approach and Landing Simulator (GRALS) at ESTEC. This rendezvous scenario will serve as Software-In-the-Loop (SIL) tests for Aut-O-MAGIC. It consists of image acquisition of a scaled Envisat model and the recreation of a representative rendezvous trajectory approaching the model.

7.1. Experimental Setup

An overview of the experimental setup can be seen in Figure 7.1. The following hardware components are required during the test campaign:

- GRALS facility:
  - Wall-mounted KUKA robotic arm;
  - VICON Motion Tracking System;
  - VICON Active Wand;
  - VICON PC;
  - GRALS PC.
- A 1:10 model of Envisat;
- Light source “stage light”;
- KUKA end-effector interface for camera mount;
- 5V battery pack;
- Azure Kinect DK (K4A) depth camera;
- Host machine for the Azure Kinect DK.

Similarly, the following software components are required:

- GRALS software:
  - (Simulink @ GRALS PC) VICON multi-object tracker;
  - (Simulink @ GRALS PC) KUKA connection block (read telemetry only);
  - (KRL @ KR C4) Robot trajectory + Simulink broadcasting script.
- (C @ Host machine K4A) trigger routine to begin and store sensor data recording.

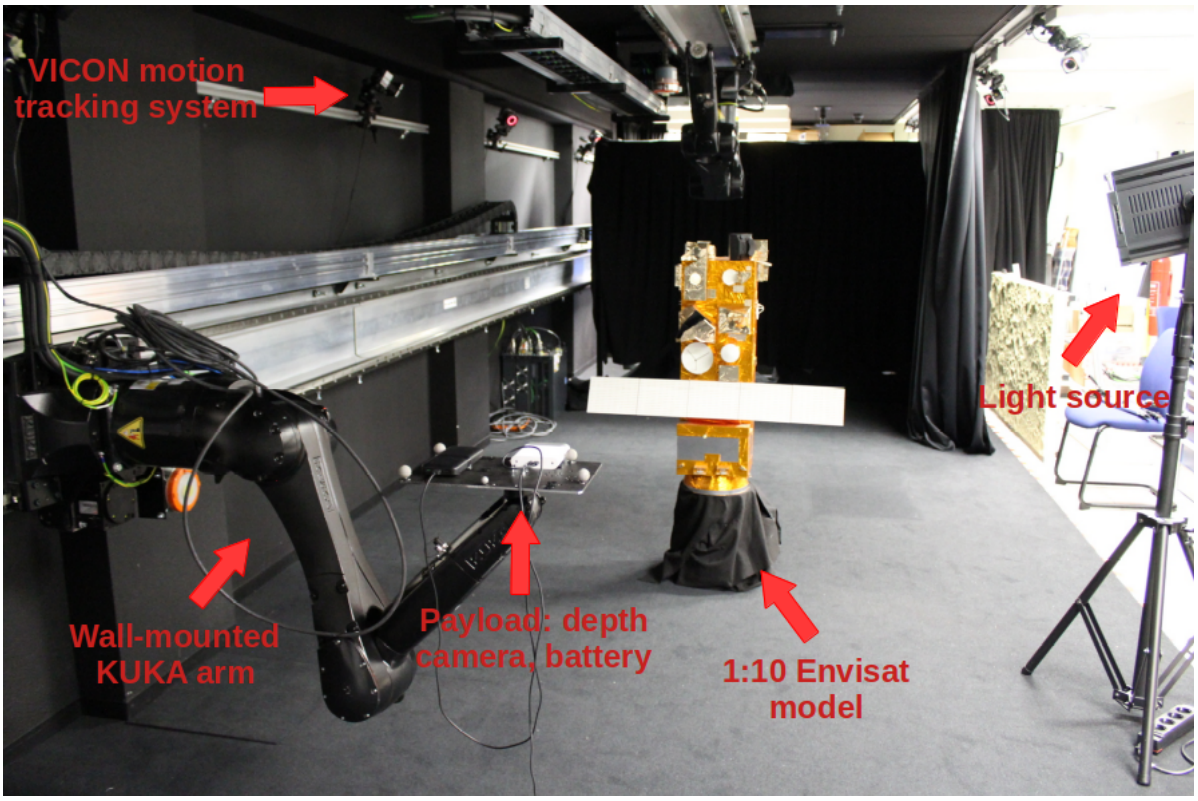


Figure 7.1: Overview of the test environment in GRALS, with the components labelled (computing equipment not in view).

7.1.1. The Orbital Robotics and GNC Laboratory and GRALS

The laboratory set-up in the Orbital Robotics and GNC Laboratory at ESTEC has multiple facilities dedicated to the study of free-floating systems. In this project the GNC Rendezvous, Approach and Landing Simulator (GRALS) facility will be used. GRALS is a test facility belonging to the GNC Test Facilities, consisting of hardware and software elements, making use of gantry robots to hold a GNC system payload and to simulate its movements for rendezvous and entry, descent, and landing missions in 6 degrees of freedom. The facility consists of two robotic arms (type Agilus KR10 R1100 sixx C) on linear tracks on the wall and ceiling of the Orbital Robotics and GNC Lab (namely Wall and Ceiling manipulators), with a useful length of 33m and 5.5m, respectively. The robots are controlled by their respective proprietary KR Controller

4 units (using proprietary KRL language), and a KUKA (Control) Cabinet which includes a safety PLC. An overview of the active components of GRALS is depicted in Figure 7.2.

Explicitly, GRALS was designed to be able to reproduce:

- Scaled landing trajectories for planetary or small body-missions descent phase,
- Scaled and 1:1 trajectories for planetary or small body-missions landing/touch-down phase,
- Scaled trajectories for the approach of the rendezvous phase of a mission,
- Scaled and 1:1 trajectories for the final approach and docking/berthing of the rendezvous phase of a mission.

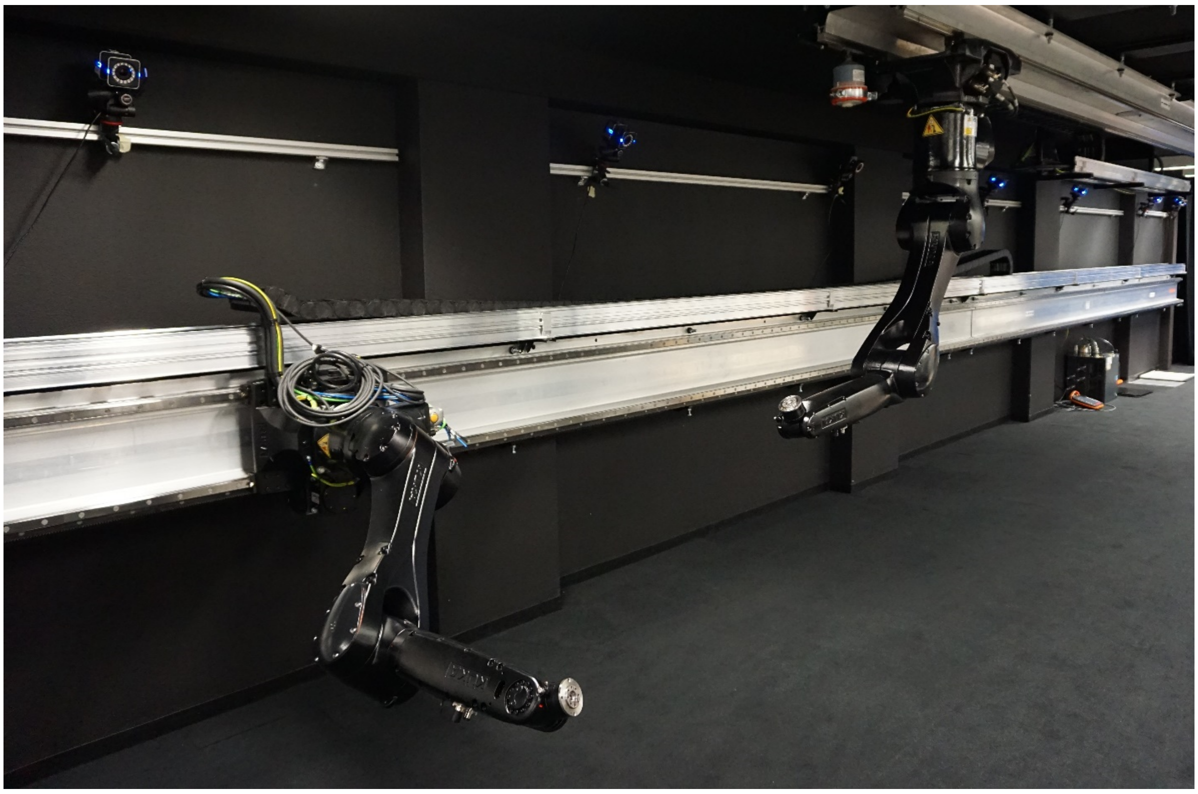


Figure 7.2: GNC Rendezvous, Approach and Landing Simulator overview.

The GRALS facility can be remotely controlled via a device on the local network over a UDP (Ethernet) connection with a latency of 4ms. Any device can be used, including a provided real-time machine to achieve consistent results.

Moreover, the GRALS facility includes a lab-wide VICON motion capture system, commonly used by industry as ground truth in experiments of this type. In the GRALS VICON setup, measurement accuracy below 1mm is routinely achieved.

In addition to the laboratory facility, a robotiq 2F-85 two-finger gripper, a Robotiq FTS-300 force-torque sensor, and one Basler acA1920-48gc GigE 50FPS 2.3MP area scan camera are available.

In terms of software, a versatile GRALS controller built in Simulink is available for use as an interface between the rendezvous and docking system being designed and the GRALS robotic arms. This GRALS controller can be compiled to the real-time machine for improved responsiveness. It is also possible to compile the rendezvous and capture controllers in the virtual machine. This requires the work on this thesis to be done in the C/C++ programming language, as the most common option, Python, has to run extrinsically and generally cannot be compiled into C.

7.1.2. Payload

The payload is depicted in Figure 7.3, and as previously mentioned consisted of the Azure Kinect DK (in white), the end-effector interface to mount the camera and the other items to the KUKA robots (black plate), the 5V battery pack to power the camera, and the host machine to command and retrieve the camera data (attached to the robot base).

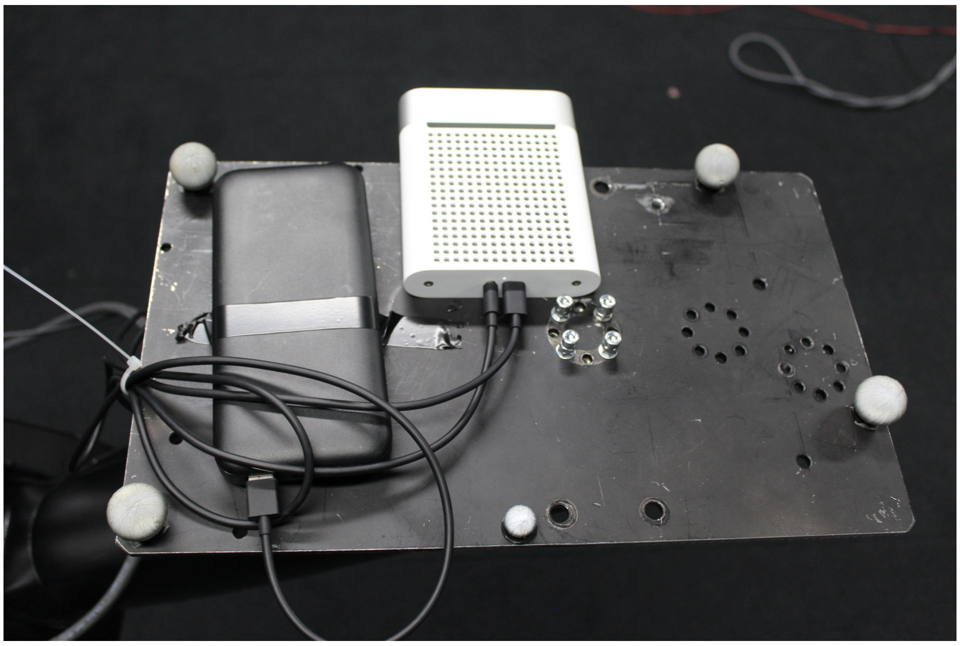


Figure 7.3: Validation test payload.

7.1.3. Target

The target Envisat 1:10 scale model is depicted in Figure 7.4. It has been assembled without the largest solar panels that are also omitted in the 3D voxel model of Envisat used in this thesis work.

7.2. Trajectory

Two trajectories were planned: a long-range (from 29.5 m until 0.1 m) linear trajectory towards the target, and a short-range orbiting trajectory (Figure 7.5). However, due to goals of this validation campaign being the determination of resolution requirements and overall correspondence of the graspable regions with the synthetic model, the short-range trajectory is ultimately not required.

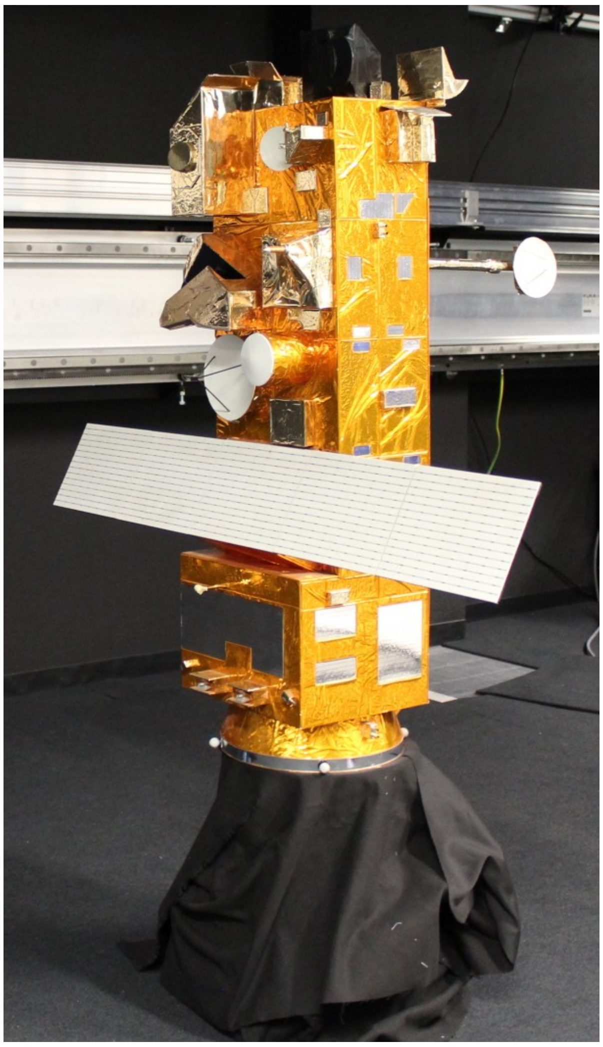


Figure 7.4: 1:10 scaled model of Envisat used in the SIL test campaign.

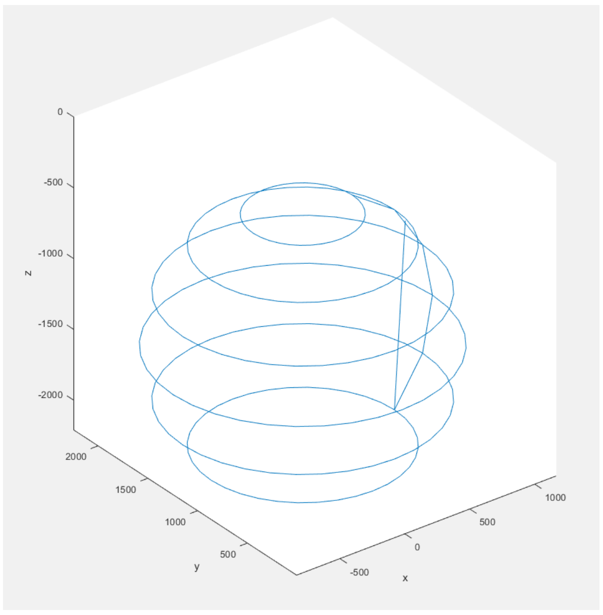


Figure 7.5: Short-range spherical trajectory executed during the SIL tests, all units in mm.

7.3. Data Collection

Data were collected following the process described in subsection 4.5.5. The sensor recorded the data on a `mkv` file that contained IR, colour, and depth channels. Moreover, telemetry from the KUKA robotic arm and VICON ensured accurate tracking.

Focusing on the analysis of the long-range linear motion to the target, in Figure 7.6, it can be observed how the settings used in the K-NN outlier removal technique ($k = 30, \sigma = 1$) were able to greatly reduce the point cloud data noise. Further noise reduction is not necessary because region of interest is the Envisat satellite, such that remaining noise in other areas can safely be disregarded.

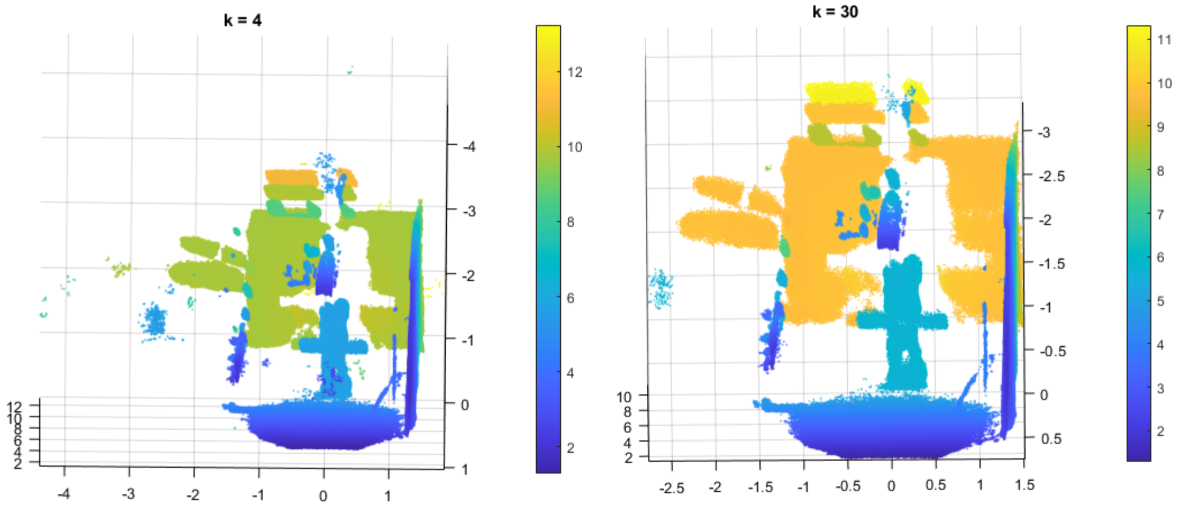


Figure 7.6: Global reconstructed point cloud compiled with the K-NN outlier removal method using the 5 nearest neighbours (left) and the 30 nearest neighbours (right). All units are in meters, with the colour gradient displaying the distance from the origin.

The entire trajectory recording spans a distance of 29.5 meters. Single point cloud frames are too sparse for grasp determination. With sensibly high resolutions, the voxel models present many gaps. With a lower resolution some of these gaps are resolved but it becomes to coarse for grasp planning. Example depictions of voxelisation at reasonable and coarse resolutions is depicted in Figure 7.7. To increase point density and surface reconstruction quality, filtering, registration, and refinement of point cloud sets is performed. Through this, 5 segments at different distances from the target are reconstructed to investigate the resolution requirements of the grasp planning algorithm on real-world data. The regions are tabulated in Table 7.1.

Table 7.1: Trajectory segments used for surface reconstruction.

Point cloud snapshot range [-]	Min distance [m]	Max distance [m]
300–349	13.16	15.48
450–499	6.07	8.39
500–549	3.70	6.02
550–599	1.34	3.66
600–610[1]	0.82	1.29

Other regions were not extracted for analysis as these already cover the entire operational envelope of the sensing system.

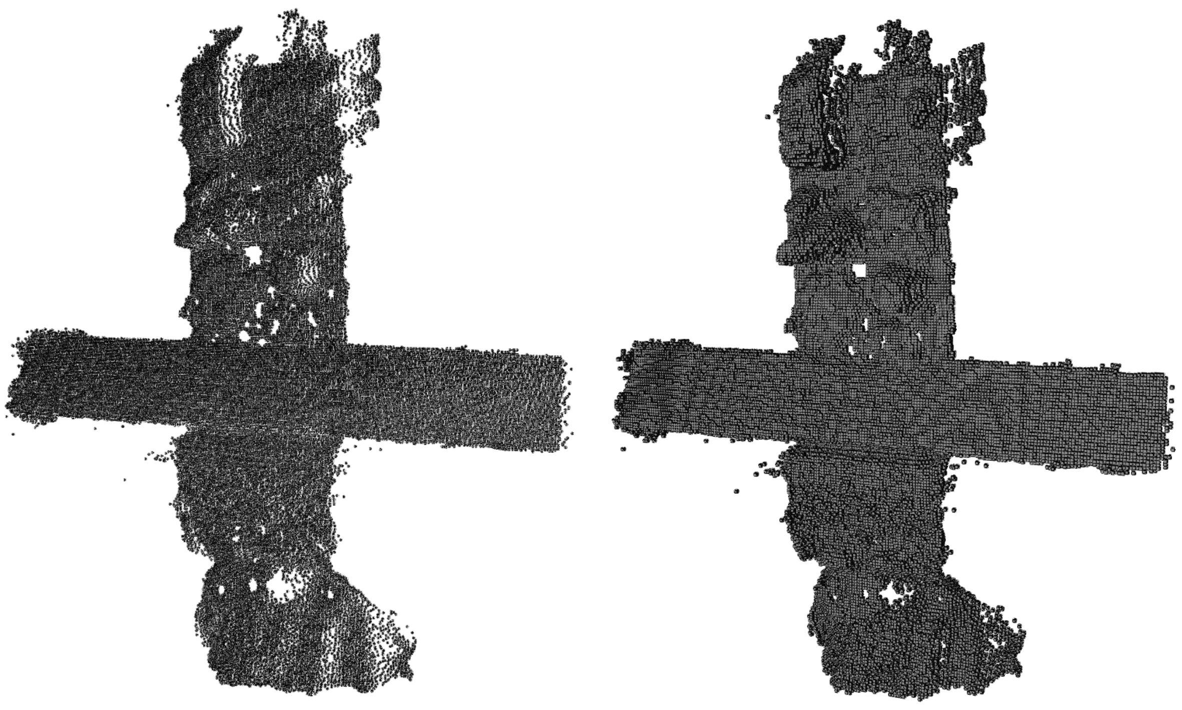


Figure 7.7: Voxel models from single point cloud snapshots of Envisat taken at a distance of 1.29 m, voxel leaf sizes of 0.002 m (left) and 0.005 m (right).

7.4. Graspability Analysis

The 5 reconstructed point cloud sets are voxelised at leaf sizes of 0.001m to 0.005m, such that there is a range of resolution models to determine the minimum resolution requirement. The voxelisation results for the different reconstructed regions is tabulated in Table 7.2.

[1]The last point cloud segment covers only 10 entries due to Envisat crossing the point where it is no longer fully observable within the camera's FOV

Table 7.2: Reconstructed voxel models from point cloud data at different distances from the target.

Distance range [m]	Leaf size [m]	Figure ref.
13.16–15.48	0.005	7.8
6.07–8.39	0.005	7.9
3.70–6.02	0.005	7.9
1.34–3.66	0.005	7.10
0.82–1.29	0.005	7.10

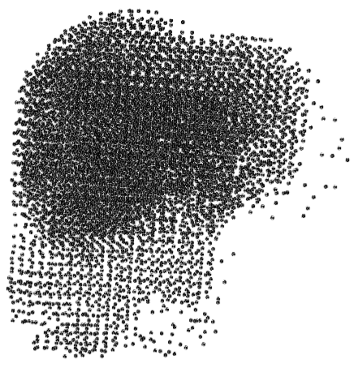


Figure 7.8: Envisat (1:10 model) reconstruction at a distance of 15.48–13.16 m, voxel leaf size of 0.005 m

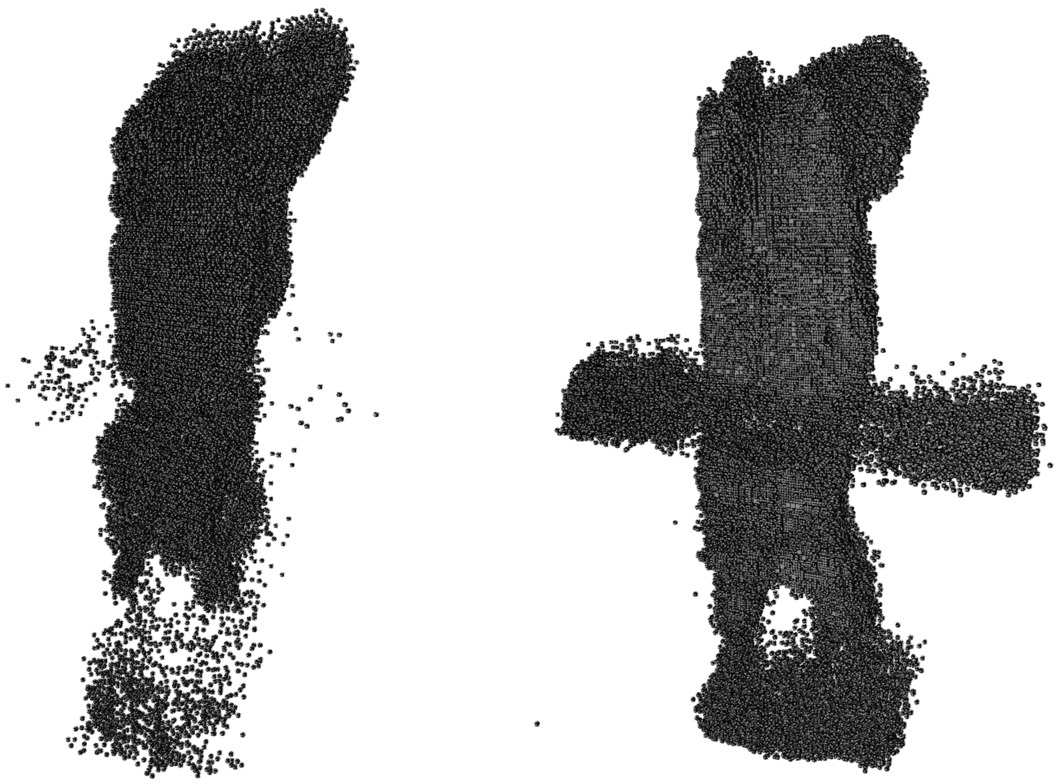


Figure 7.9: Envisat (1:10 model) reconstruction at a distance of 8.39–6.07 m (left), and 6.02–3.70 m (right), voxel leaf size of 0.005 m.

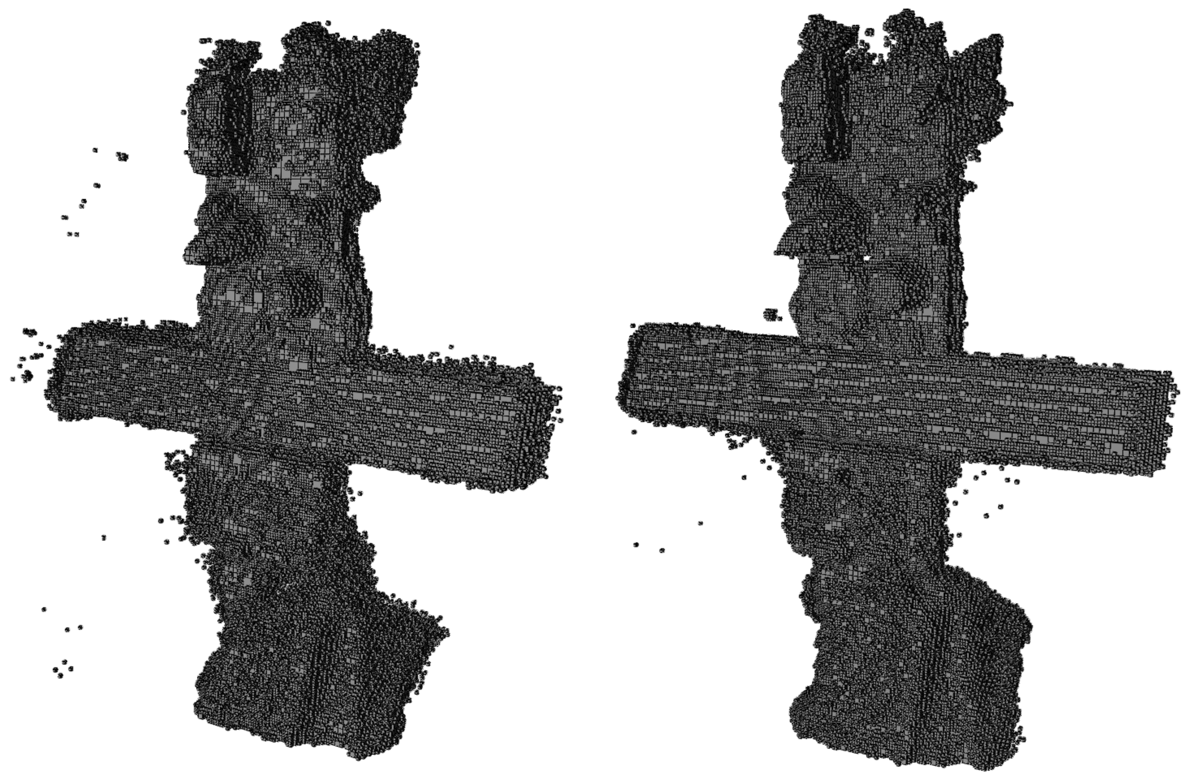


Figure 7.10: Envisat (1:10 model) reconstruction at a distance of 3.66–1.34 m (left), and 1.29–0.82 m (right), voxel leaf size of 0.005 m.

With the Envisat model reconstructed, global grasp analyses are performed to determine the robustness of the graspability analysis using as comparison the noise-free Envisat synthetic model used in section 6.1, as well as the minimum resolution required for meaningful results. During grasping analysis, the observed Envisat object is upscaled by a factor of 10 to match the dimensions of the real Envisat satellite used in the synthetic analyses. With that, observing the 1:10 scale mock-up with a leaf size of 0.002m roughly corresponds to the leaf size used for synthetic analysis (0.019m). The grasp analyses are performed using method #7: voxel superimposition and grasping surface coplanarity, which was shown to deliver good performance during verification. The results are tabulated in Table 7.3.

Table 7.3: Results of the validation analyses of Envisat at different leaf sizes (resolutions). Envisat was reconstructed from a single point cloud snapshot at 1.29 m distance and a set of point cloud snapshots at 1.29–0.82 m distance.

Leaf size [m]	Figure ref.	Comments
0.001	N/A	The decompressed model requires more than 16GB of RAM, unable to analyse
0.002	7.11	Analysis from singular point cloud is too sparse; analysis from set of clouds too noisy
0.003	7.12	Analysis from singular point cloud is too sparse; analysis from set of clouds shows graspability patters comparable to synthetic analyses, albeit requires further noise reduction
0.004	7.13	Singular point cloud has acceptable point density and graspability patters are comparable to synthetic analyses, but clearly displays grainy results; set of point clouds has less grainy results than single snapshot, low voxel resolution incorrectly leads to varying graspability results along the edges of circular surfaces such as the antennae
0.005	7.14	Resolution is too coarse for graspability analysis given the $8.4 \times 3.6 \times 1.9$cm gripper used in this work[2]

It can be seen how it is difficult to robustly predict grasps in the target Envisat model with the defined upscaling of the captured model. For illustration purposes, an analysis is also performed without upscaling the model—keeping the metric scale as-is. This yields the results depicted in Figure 7.15, which clearly show much better performance. In Figure 7.16 and Figure 7.17, close-up views of the global graspability map around the best grasp candidate are depicted.

[2]The envisat model was scaled up by a factor of 10 during analysis to match the size of the synthetic model and allow using the same gripper size. This is equivalent to using a $0.84 \times 0.36 \times 0.19$cm gripper with the unscaled model

7.5. Conclusions

This validation campaign demonstrated that grasp planning with analytical methods is very sensitive to noise. However, clear graspability patters were still observable which suggest that, with superb noise reduction and model reconstruction pipelines, reproducible results as the ones obtained in the verification analyses with synthetic models are possible. This campaign has also shown how generating the global graspability maps using larger grippers proves to be much more robust to noise. These observations suggest that in order to increase robustness to noise the resolution-to-gripper-width ratio shall be increased.

It has been observed that in order to perform global grasp analyses of the target models, the resolution (leaf size) has to be less than half (1/2) the grasping width of the gripper employed. This draws clear parallels to the characteristic Nyquist frequency of samplers in signal processing. With a sampling (resolution) wider than half the grasping width, aliasing is encountered and potential grasping candidates become scarce and inconsistent.

Lastly, it is worth noting that it can be observed how, using the Microsoft Azure Kinect DK (K4A), it was possible to obtain an accurate image of the target model at a resolution of 0.005m from a distance of 3.6m, which largely exceeds the typical close proximity rendezvous sensor accuracy requirements outlined from a study of previous missions [15], [22]. This suggests that RGB-D sensors are a promising for accurate mapping in space. However, these sensors are vulnerable to IR noise due to their active IR depth sensing technology.

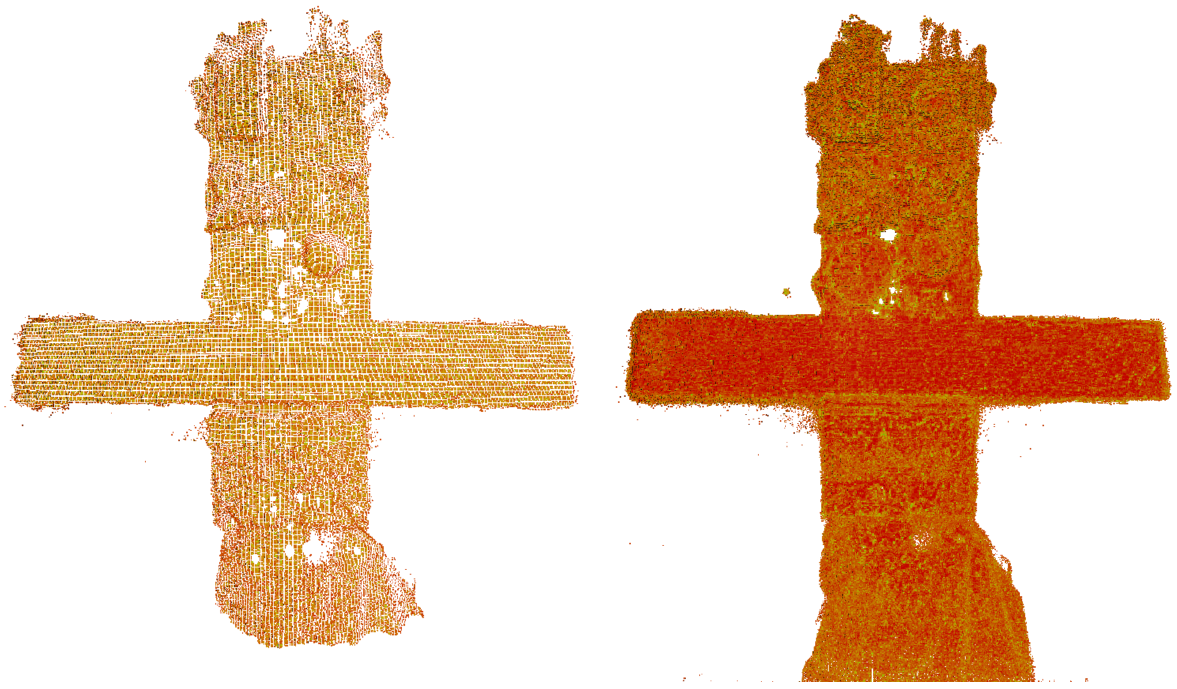


Figure 7.11: Global grasp planning of Envisat (1:10 model) reconstructed from a single point cloud snapshot at a distance of 1.29 m (left) and 10 snapshots at a distance of 1.29–0.82 m (right), voxel leaf size of 0.002 m.

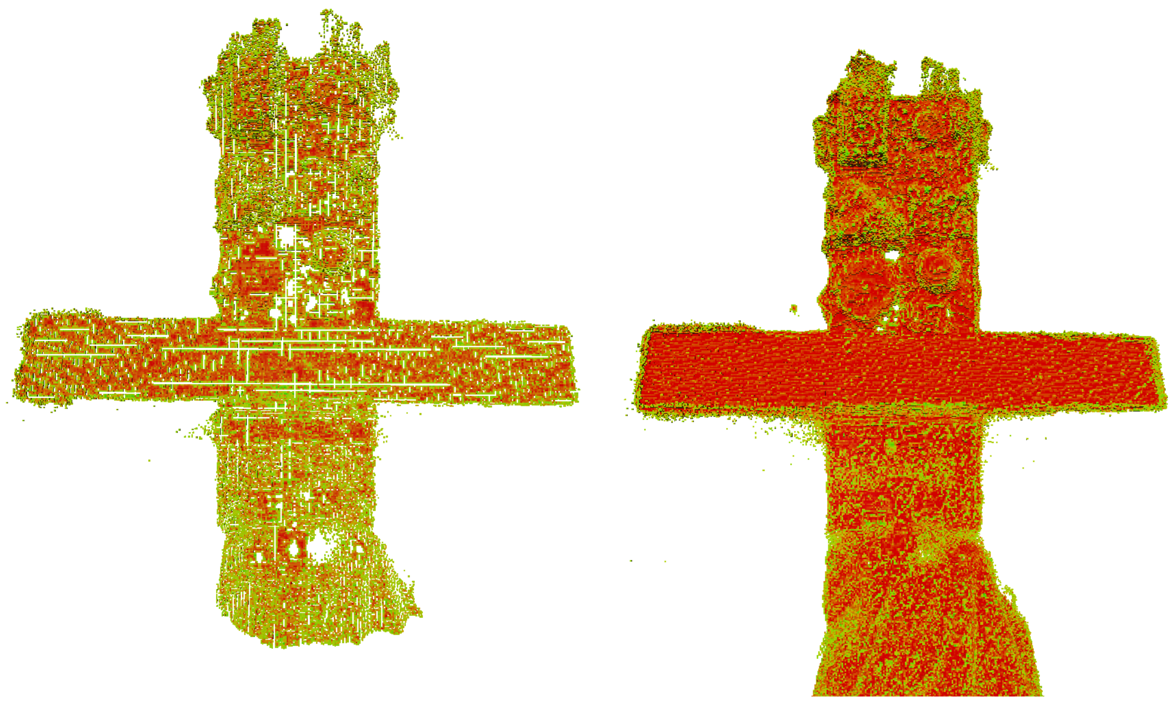


Figure 7.12: Global grasp planning of Envisat (1:10 model) reconstructed from a single point cloud snapshot at a distance of 1.29 m (left) and 10 snapshots at a distance of 1.29–0.82 m (right), voxel leaf size of 0.003 m.

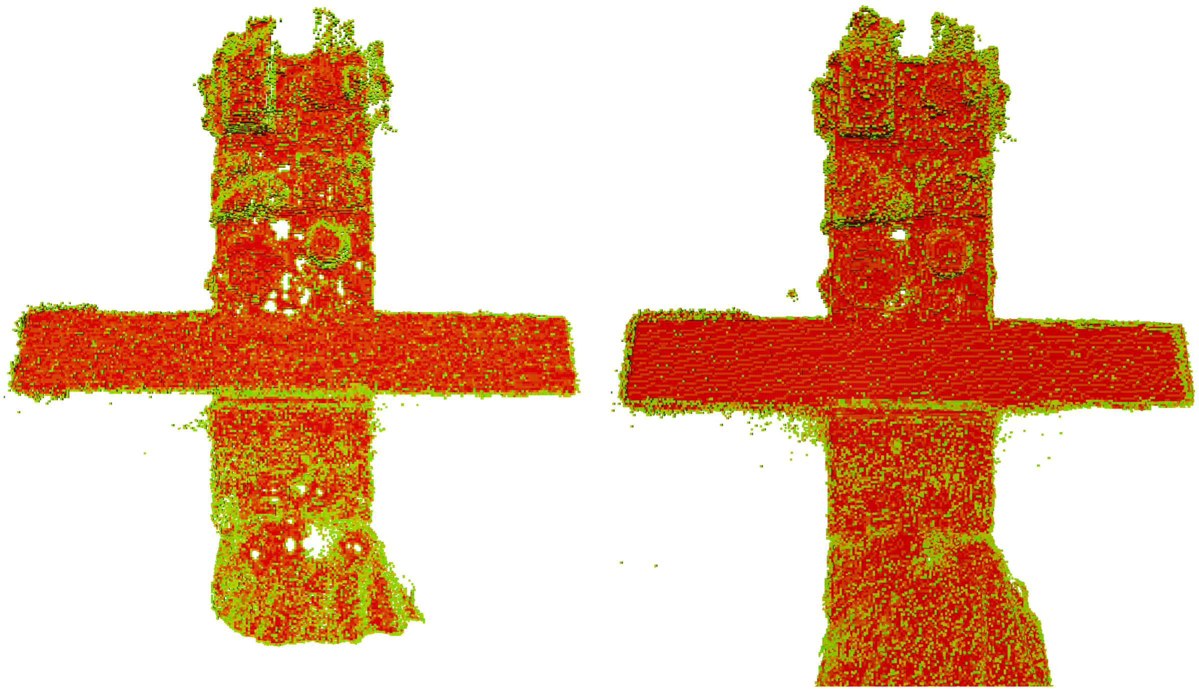


Figure 7.13: Global grasp planning of Envisat (1:10 model) reconstructed from a single point cloud snapshot at a distance of 1.29m (left) and 10 snapshots at a distance of 1.29–0.82m (right), voxel leaf size of 0.004m

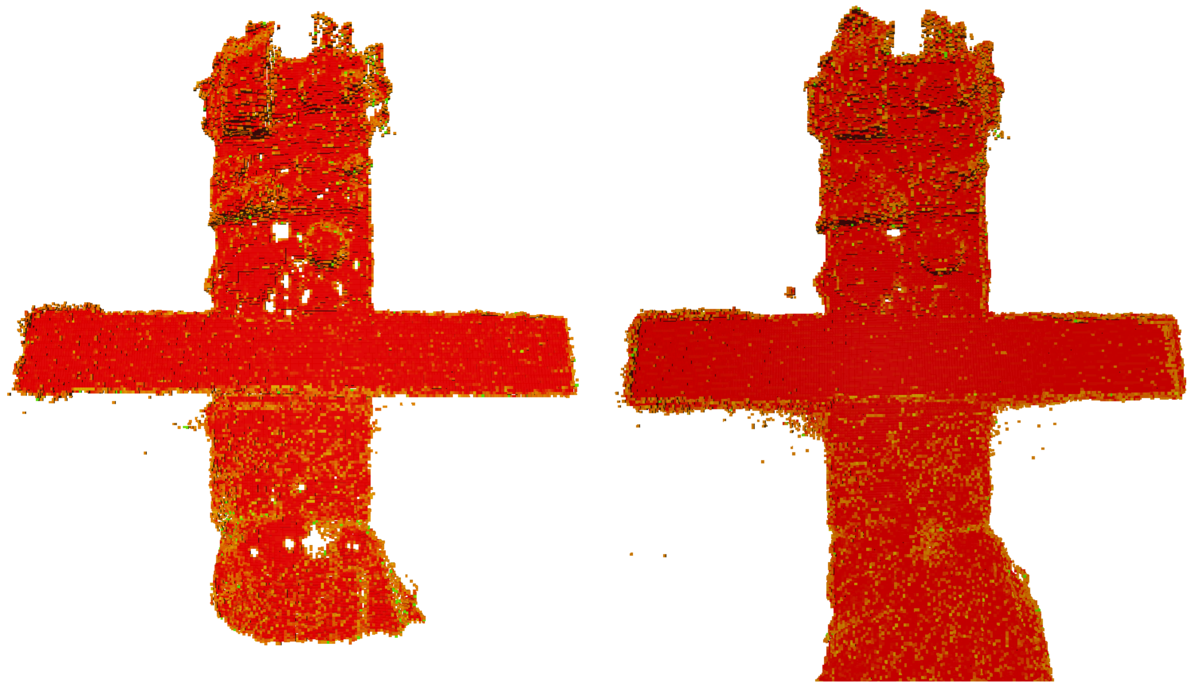


Figure 7.14: Global grasp planning of Envisat (1:10 model) reconstructed from a single point cloud snapshot at a distance of 1.29 m (left) and 10 snapshots at a distance of 1.29–0.82 m (right), voxel leaf size of 0.005 m.

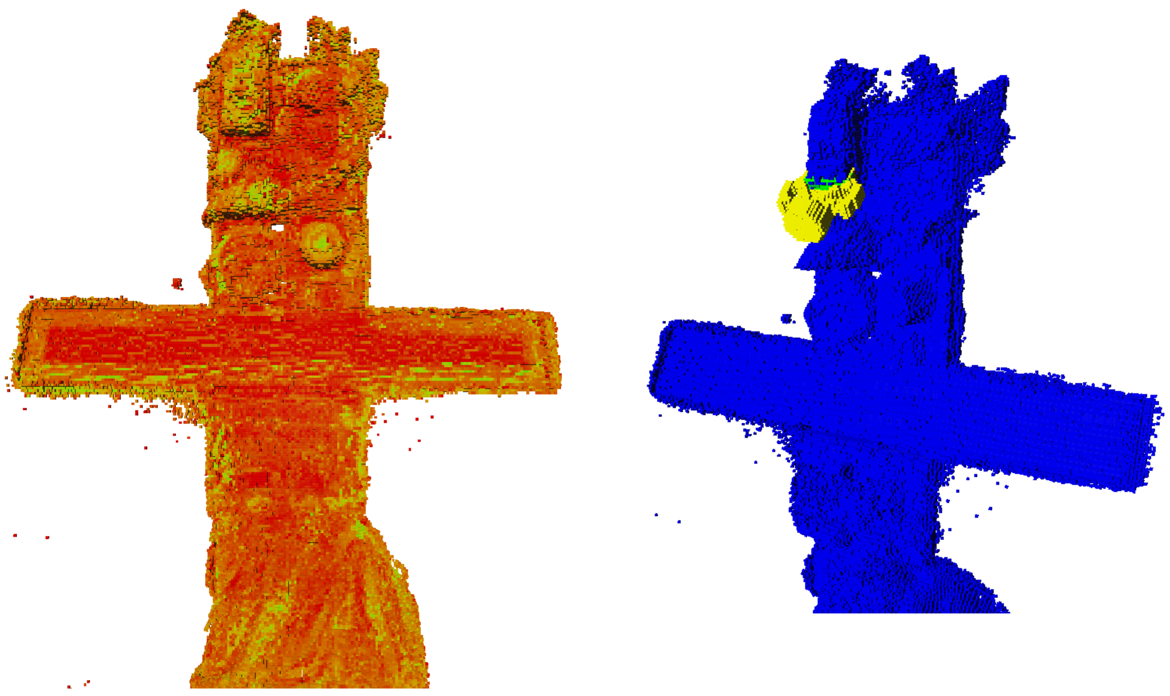


Figure 7.15: Global grasp planning and best grasp candidate of Envisat without upscaling and maintaining the standard gripper width of 8.4 cm. Reconstructed from 10 snapshots at a distance of 1.29–0.82 m, voxel leaf size of 0.005 m.

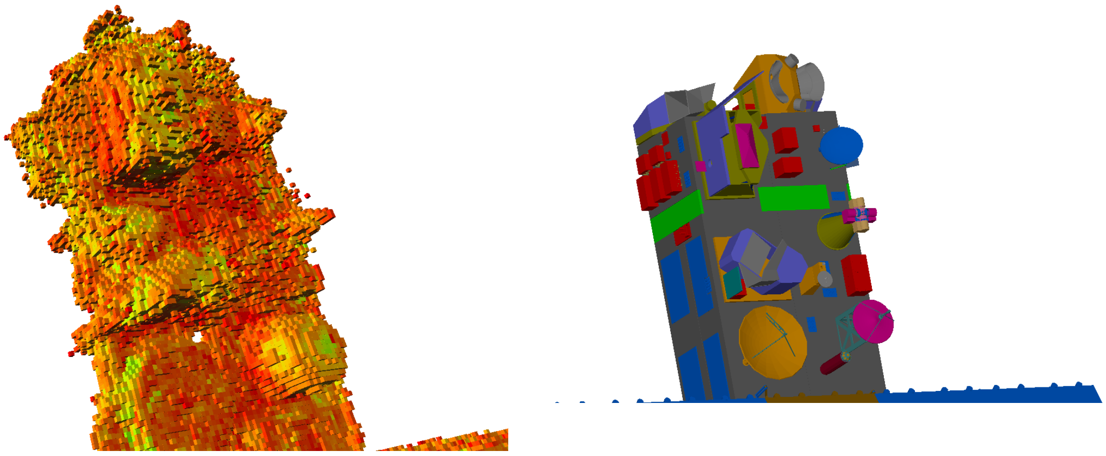


Figure 7.16: Most graspable region of the unscaled Envisat model analysis (viewpoint 1).

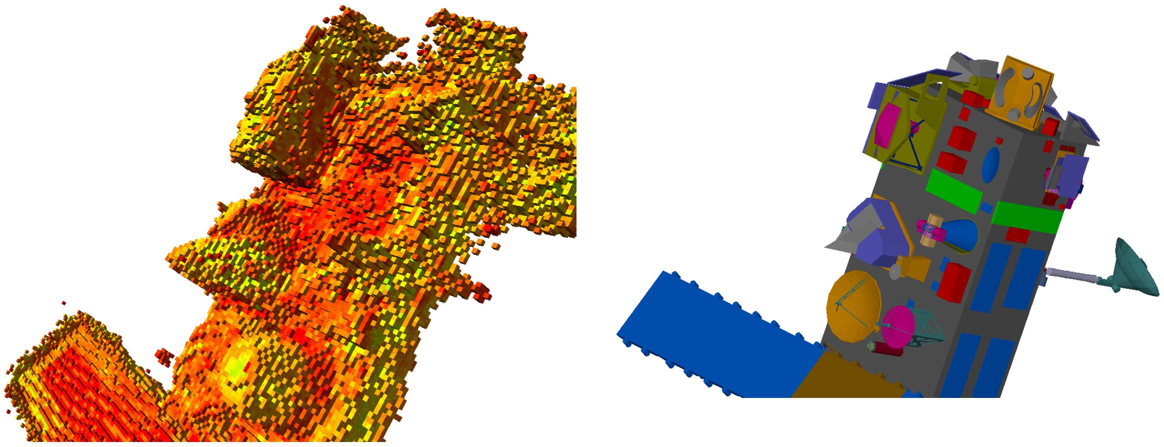


Figure 7.17: Most graspable region of the unscaled Envisat model analysis (viewpoint 2).

IV

Conclusions and Recommendations

8 Requirements Compliance

This chapter describes the compliance of the Aut-O-MAGIC design solution with the technical requirements defined. In the table below, the technical requirements compliance list is tabulated.

Table 8.1: Technical requirements compliance list.

ID	Status	Rationale
SYS-REQ-1	OK	Verified by test: The system generates a 3D reconstruction of the scene with acceptable noise reduction.
SYS-REQ-2	OK	Verified by analysis: grasp candidates are generated at every voxel node, and there is sufficient population of high-scoring candidates.
SYS-REQ-3	OK	Verified by test: Grasp candidates are quantitatively ranked by the grasp quality value $[0 \dots 1]$ and visually by their colour-mapping.
SYS-REQ-4	OK	Verified by test: 3D voxeliser works and its operation is documented in section A.2
SYS-REQ-5	OK	Verified by test: Colour-mapping of the grasp quality value of all nodes is provided and demonstrated to be working.
SYS-REQ-6	OK	Verified by test: 3D model to depth map utility is provided and utilised successfully.
SYS-REQ-7	OK	Verified by inspection: 3D models with a resolution below 2.5 cm generated successfully for both synthetic data and sensor data.
SYS-REQ-8	OK	Verified by inspection: Validation runs cover envelope 0.1–29.5 m.

SYS-REQ-9	OK	Verified by analysis: Grasp instantiation is performed in a deterministic fashion, employing semi-optimised instantiation that iterates over all surface nodes, pointing the gripper along the surface normal vector.
SW-REQ-SA-1	OK	Verified by test: Driver correctly outputs point cloud snapshots from the sensor recording.
SW-REQ-SA-2	OK	Verified by test: The system registers point clouds against a reconstructed scene with acceptable noise reduction.
SW-REQ-SA-3	OK	Verified by test: The system compiles a scene reconstruction of multiple point cloud snapshots.
SW-REQ-SA-4	OK	Verified by test: The system converts a point cloud into a binary voxel tree with user-defined resolutions.
SW-REQ-VOX-1	OK	Verified by test: process and example detailed in section A.2.
SW-REQ-GP-1	OK	Verified by test: 3D voxel models are imported from `.bt` files and successfully analysed.
SW-REQ-GP-2	OK	Verified by analysis: Grasping information is stored back to the trees, demonstrated by colour mapping in colour trees, which would otherwise not be possible.
SW-REQ-GP-3	NOK	The system does not iteratively analyse *all* nodes in the octree, but all nodes on the surface. This is done for performance considerations.
SW-REQ-GP-4	OK	Verified by inspection.
SW-REQ-GP-5	OK	Verified by inspection.
SW-REQ-GP-6	OK	Verified by inspection.
SW-REQ-GP-7	OK	Verified by test: Output files are written with data integrity.
SW-REQ-DEPTH-1	OK	Verified by test: 3D model visualisation with custom camera location successfully generates depth image.
SW-REQ-DEPTH-2	OK	Verified by test: Camera adjuster GUI is present.
SW-REQ-DEPTH-3	OK	Verified by analysis: Depth array is generated and raw cell values show correct trends.
SW-REQ-DEPTH-4	OK	Verified by analysis: Segmentation mask is a binary casting from the depth array, which was verified to work.
SW-REQ-DEPTH-5	OK	Verified by test: Depth image is generated from a user-defined viewpoint.
SW-REQ-DEPTH-6	OK	Verified by test: Output files are written with data integrity.

9 Conclusions

This chapter discusses how the research questions of this thesis have been answered. On a high level, the performance of current analytic autonomous grasp planning methods for terrestrial applications can be improved to enable robust on-orbit grasp planning of uncooperative novel satellites by embracing the unique opportunities of the space environment with the generation of occlusion-free, time-invariant, global graspability maps of the target novel satellites prior to gripper interaction.

The methods proposed in this thesis span the two primary principles of *mass* under the gripper grasping plates, and *coplanar* contact surface on the target object. By analysing them separately and in a weighed manner, it has been observed how the method which combines voxel superimposition and target contact surface uniformity performs satisfactorily. On all the adversarial objects analysed grasp quality scores of up to 0.9 out of 1 were achieved and, although there is difficulty validating this score without physical grasping test campaigns, the provided visualisation tools based on geometric interactivity suggest promising sustainment of simulation performance in real-world operations. Furthermore, ad-hoc comparison of the grasp planning solution developed in this thesis against the solutions by the state-of-the-art Dex-Net 4.0 grasp planning algorithm, displayed clear agreement of results. In some cases, it was also observable how the algorithm designed benefited from the additional information available in 3D maps, and was able to translate that information into better grasp candidates than what was possible with Dex-Net, which relied on depth images.

Successful implementation of global graspability maps as proposed in this work will provide greatly enhanced on-orbit capabilities. As it has been discussed with rover traverses, supplying detailed traversability maps can greatly improve the autonomy of the rover's traverse and enable globally-optimised sets of manoeuvres minimising travel time, motor effort, risk, and many more. Global graspability maps for on-orbit servicers would unlock an exciting set of complex manipulability operations that were previously not possible.

Below, the central research question is re-stated for clarity, together with the sub-questions and their outcomes which, overall, answer the central research question.

> How can the performance of current analytic autonomous grasp planning methods for terrestrial applications be improved to enable robust on-orbit grasp planning of uncooperative novel satellites?

1. What requirements are relevant for evaluating the performance of an autonomous space grasp planning system?

(a) Which resolution is required to achieve good results?

Analytical grasp planning methods are more susceptible to noise than their empirical counterparts. This sensitivity to noise has been clearly demonstrated in the validation tests, and can be ameliorated with thorough outlier removal algorithms and accurate surface reconstruction. It is demonstrated that, with noise mitigation measures in place, the global grasp quality can be determined reliably if the resolution (leaf size) is less than half (1/2) the grasping width of the gripper employed. With a sampling (resolution) wider than half the grasping width, aliasing is encountered and potential grasping candidates become scarce and inconsistent.

(b) What noise levels does the system need to reliably operate under?

In the validation campaign it was demonstrated that grasp planning with analytical methods is very sensitive to noise. This campaign has also shown how generating the global graspability maps using larger grippers proves to be much more robust to noise. These observations suggest that in order to increase robustness to noise, the resolution-to-gripper-width ratio shall be increased beyond the minimum half grasping width previously determined. The specific ratio needed for an application will depend on the particular noise levels in the environment, and what 3D imaging sensor is used. In this scenario, using an RGB-D camera, the infrared noise is the constraining factor.

(c) What constitutes a successful grasp?

Grasp quality metrics have been defined in multiple ways in this work, in order to compare them and realise the most truthful representation of grasp success. It was observed how the metrics composed of voxel superimposition scores between the target and gripper objects, together with a measure of target surface contact points coplanarity, performed best. With iterative weight tuning, it was possible to achieve grasp quality candidates that are verifiable by inspection. A successful grasp can be defined to be any candidate that has a grasp quality score above a certain threshold, computed using those grasp quality metrics. In practise, grasp candidates with a score of 0.6 or more with the 1/10 reward/penalty tuning and a bin saturation of 50, provided successful grasps reviewed by inspection.

2. Which validation procedures most truthfully recreate the operating conditions of the system in space?

Validation procedures that include a rendezvous phase from a distance at which the chaser can safely map the target object, paired with a light source of different spectrum as the sun, will render truthful simulations of the operating conditions of the system in space. Reliability tests can be executed on a frictionless surface (such as an air-bed) to quantify the grasp reliability of the system.

3. How can geometric matching using 3D maps provide a performance advantage compared to image-based analyses?

As it has been demonstrated throughout this thesis work, terrestrial grasp planning sensors frequently rely on target geometric information from a single viewpoint. This is due to the constrained nature of grasping operations on Earth, where gravity is a factor and there is a significant trade-off to perform between resolving a larger amount of the target object surface and saving time and effort and attempting the grasp with the limited information available from a single viewpoint. In space, however, this is not the case. The free-floating dynamics of space enable chaser satellites to perform proximity operations around target vehicles and assess all potential viewpoints for very limit expense.

(a) What are the limitations of image-based grasp planning?

Image based grasp planning solutions cannot reconstruct surface topology and hence is limited to analyse the surface graspability in the axial direction. This forgoes analysis of the contact points of the gripper plates on the target object. These contact points provide significant insights into the graspability of the surface as can compute the homogeneity of the surface and accordingly penalise cavities.

(b) How does the computational efficiency of both methods compare?

Image-based grasp planning methods do not require expensive pre-processing of the sensor data like 3D map-based analyses. For 3D map-based analyses, the snapshots have to be registered against a global 3D map of the target, while image-based methods directly interface with the images acquired from the camera sensor. However, if the goal of performing a global analysis is shared by both methods, image-based methods impose stricter real-time requirements on the system, as the images from the sensor instrument cannot be stored against a global 3D map for later analyses. In this work, a full surface analysis of Envisat, analysing 10^6 candidates, took between 1675 s (28 min) and 4029 s (67 min), depending on the grasp quality metrics used. The analyses were performed on a single-threaded environment running at 2.8 GHz. Meanwhile, an analysis of a single grasp candidate by Dex-Net took approximately 5 s to complete, which would mean that Dex-Net could analyse between 335 and 805 images during the execution of the Aut-O-MAGIC algorithm. These execution times are not comparable, however, due to the initialisation overhead suffered Dex-Net on every iteration, and the fact that 3D map registration was performed ahead-of-time in Aut-O-MAGIC.

(c) Do less-occluded 3D maps provide a robust knowledge advantage over single-view depth images?

It has been observed in the comparison between this thesis' grasp planning solution and the depth image-based Dex-Net 4.0 solution how the image-based method has a preference for grasping locations where the grippers can lie completely outside of the target object's bounds (free space). This can be particularly limiting in space, where the usual size of space objects is large, and an observer is frequently required to observe grasping points of interest that have further structure behind them. Precise determination of surface topology using 3D maps allows planning grasps that are fully within the object bounds with more confidence.

The source code of this thesis work can be found on GitHub, under the Aut-O-MAGIC organisation[1]. The voxel tree classes for grippers and targets, the grasp planning engine, the depth map generator, and the scene reconstruction engine are released publicly. Other tools such as the Azure Kinect DK driver have not been made public as they are not generalisable.

[1]`https://github.com/aut-o-magic`

10 Recommendations for Future Work

This chapter itemises recommendations for future work that emerged during the execution of this thesis work. It includes suggestions for improving the grasping algorithm designed in this thesis work, as well as higher level recommendations on other promising design approaches.

- The current grasp candidate instantiation method is optimised for thick structures. Without instantiating inner nodes, thin structures are observed to be missing promising grasp poses. In this work, inner node candidate instantiation was not performed due to significant performance penalties. A candidate instantiation method that accommodated for this candidacy underperformance without penalising performance to the degree that was experienced in this work would further cover the potential envelope of grasp candidates.
- Integrating target object mass properties in the grasp planning algorithm would provide insight into the torque that can be safely applied on the target contact points without destabilising the satellite.
- Integration and triage of target surface textures could give additional insight into surfaces' toughness. This would avoid grasping fragile elements such as solar arrays or thin reflective protective layers.
- Physical grasping tests performed on a frictionless surface (air bed) would provide highly insightful results on the graspability of objects in space.
- An Octree implementation using an SQL database has potential to provide very high search speed for nodes within the tree, and overall increase the efficiency of 3D map-based analyses.
- A grasping simulator using the physics-based engine MuJoCo would likely provide accurate first estimates on the performance of the grasp planning algorithm, such that quantitative analyses can also be performed.

V

Appendices

A System Operation

This appendix describes the operating methodology of the different systems, including UML state diagrams, UML sequence diagrams, and sample input/outputs of the systems. Because this is a tutorial of operation, rather than part of the design process, this is included as an appendix.

A.1. Grasp Planning Engine

Execution of the grasp planning engine is done through the designed `gp_node` entry point. The entry point has the options and accompanying arguments tabulated in Table A.1. For ease of understanding, a UML state diagram for the global grasp analysis option is depicted in Figure A.1. The state diagram shows as entry point the invocation of the executable in the system, which leads to an uninitialised state that needs to be populated. Then, gripper and target 3D voxel trees are instantiated and, if no errors occur during said process (i.e. invalid format, empty tree) the program transitions to a *loaded* state. From that point on, it parses the grasp planning algorithm selection that was supplied to it to determine which method to use, and then proceeds to the *global analyser* state where the global grasp quality of the target object is analysed, using the gripper object as a mold for geometric fitting. Lastly, once it finished, it moves onto the *finished analysis* state. There, it writes all outputs generated to file and proceeds to terminate program execution.

Table A.1: Command-Line Interface arguments for the Grasp Planning Engine entry point.

Option <Arguments>	Description
`—-help`	Print this message. Options declared will be serially executed in the order listed here
`—-target <tree_path>`	Target tree filepath
`—-gripper <tree_path>`	Gripper tree filepath
`—-use_simple_gripper`	Use a simple gripper model instead of importing a gripper tree
`—-write_target <save_path{=target.ot}>`	Write target octree to file
`—-write_gripper <save_path{=gripper.ot}>`	Write gripper octree to file
`—-write_color_target <save_path{=colortree_target.ot}>`	Write to file ColorOcTree version of target octree
`—-write_color_gripper <save_path{=colortree_gripper.ot}>`	Write to file ColorOcTree version of gripper octree
`—-write_surface_normals_density`	Visualise the surface normals density of the target tree
`—-gp_algorithm <idx>`	Select grasp planning algorithm to use in idx range [1–7]
`—-global_analysis`	Perform a global graspability analysis
`—-local_analysis=<{x,y,z}>`	Perform a local analysis at a defined target 3D point. Pass arg with no spaces and with equal sign (i.e. `–local_analysis={x,y,z}`) [m]

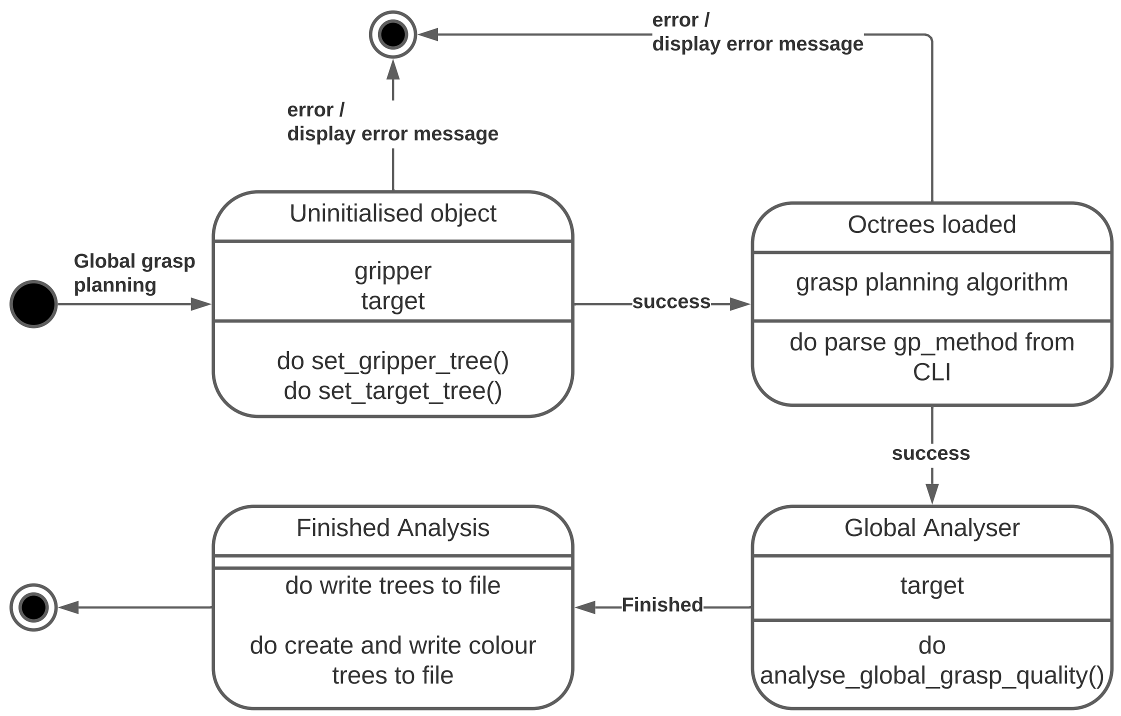


Figure A.1: UML State Diagram for the Global Grasp Planner in the case of a global grasp analysis.

A.2. 3D Model Voxelisation

To convert 3D models (usually `.OBJ` or `.STEP` files) into octrees, a series of steps are necessary. The transformation from 3D model to voxel model for Envisat is depicted in Figure A.2 and Figure A.3, with the steps documented below and the overall pipeline listed in Script A.1:

1. Convert 3D model into standard Wavefront `.OBJ` file: because models can be provided in a range of formats, it is convenient to hegemonise the input format to ensure repeatability and quality. The Wavefront 3D model format was chosen because of its widespread usage and demonstrated ability to perform well in the pipeline. A good free program to convert 3D models into Wavefront models is CAD Assistant[1].

2. Rasterise Wavefront `.OBJ` file to binvox `.binvox` model: binvox is a utility that reads 3D model files, converts them into a binary 3D voxel grid, and writes them to a resulting voxel file [71], [72]. It has a series of methods for carving the resulting voxel grid, and below some tips are provided on its usage.

3. Convert binvox model to Octomap's binary tree model `.bt`: the OctoMap package includes a utility to translate binary voxel trees from the `.binvox` format to the `.bt` format. This last step allows successfully importing the 3D voxel model into the grasp planning pipeline for analysis.

[1]`https://www.opencascade.com/products/cad-assistant/`

Script A.1: 3D model voxeliser pipeline (pseudo-Bash commands)

```
    # model.* ---> model.obj
    ./cad_assistant_1.5.0_2020-12-28_lin64.appimage

    # ---> model.binvox
    ./binvox -d 1024 -c -v -dc -pb -down -dmin 1 <model.obj>

    # ---> model.binvox.bt
    binvox2bt <model.binvox>

    # Visualise the final result
    octovis <model.binvox.bt>
```

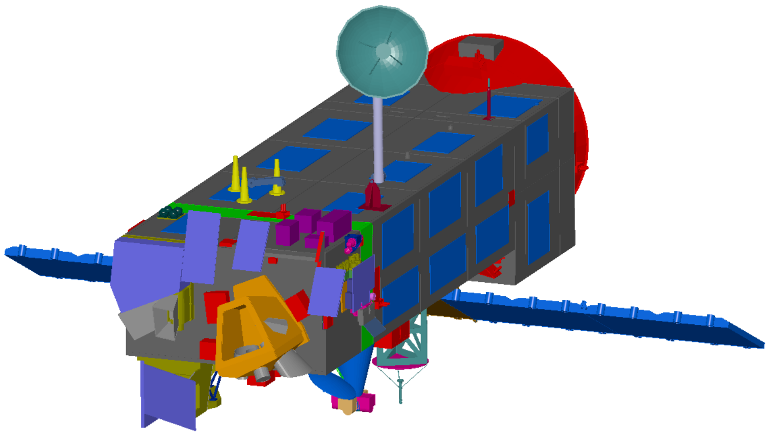


Figure A.2: Envisat 3D Wavefront model.

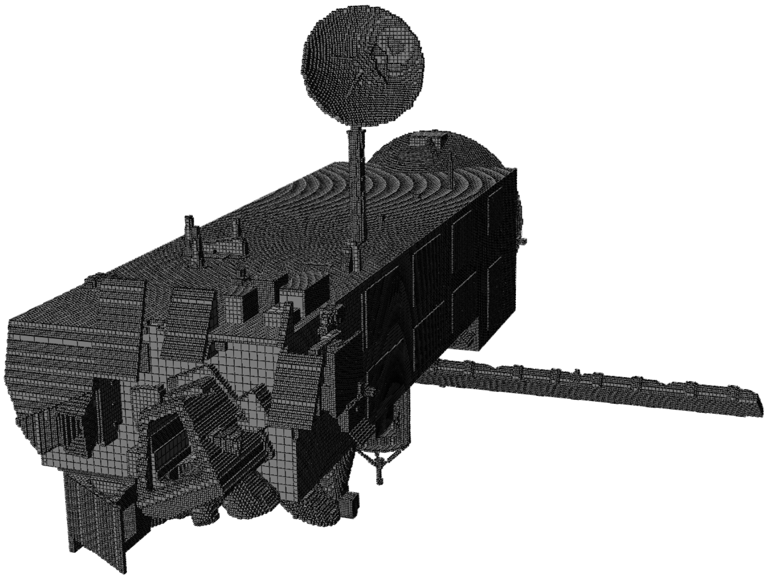


Figure A.3: Envisat binary tree (voxel) model with a leaf size of 0.0196043 m.

Some useful tips to accomplish a successful rasterisation of the 3D model file are:

- Using the off-screen buffer (`-pb`): artifacts were frequently encountered when using the on-screen buffer, but these were remediated with the alternative buffering method;

- Obtaining the desired voxel leaf size: because the resolution of the voxel tree depends on the size of the 3D model, it is more convenient to accomplish a set metric leaf size by iteration:

  1. Generate an initial binary tree (`.bt`) and open it with OctoMap's visualisation tool *octovis*;
  2. Using the layer slider in octovis, determine which layer yields the desired metric leaf size;
  3. Then, back in *binvox*, use the command modifier `-down` to downscale it by one layer from the previous maximum layer. The command modifier can be called multiple times and hence achieve any number of layers downscaling.

- Dealing with thin structures: there are multiple thickening techniques to encourage not carving away voxels from thin structures during rasterisation. It is recommended to use a water-tight 3D model. Then, the following combination of program options yield the best results:

  – `d 1024` (limited to `-d 512` in newer versions): render in maximum voxel grid size possible (will downsample later);
  – `c`: z-buffer based carving method only (keeps interior solid);
  – `v`: z-buffer based parity voting method;
  – `dc`: dilated carving, stop carving 1 voxel before intersection;
  – `pb`: use off-screen pbuffer instead of on-screen window;
  – `down`: downsample voxels by a factor of 2 in each dimension until the desired leaf size (can be used multiple times);
  – `dmin 1`: when downsampling using `-down`, destination voxel is on if >= <nr> child voxels are occupied (default 4/8). Using nr=1 allows very-thin structures to be recognised.

A.3. 3D Model to Depth Map

The utility's workings are described in the UML Sequence Diagram depicted in Figure A.4. The user has the responsibilities of:

- Launch the tool, providing as `arg` the relative path to the 3D model file to load. Optionally, a second argument can be provided to change the scale of the depth image. This does not change the values of the depth array in `npy` format, as these follow the metric scale and are hence not colour-intensity coded.

- Adjust the camera view from the GUI window that will appear to the location from which the depth image should be taken.

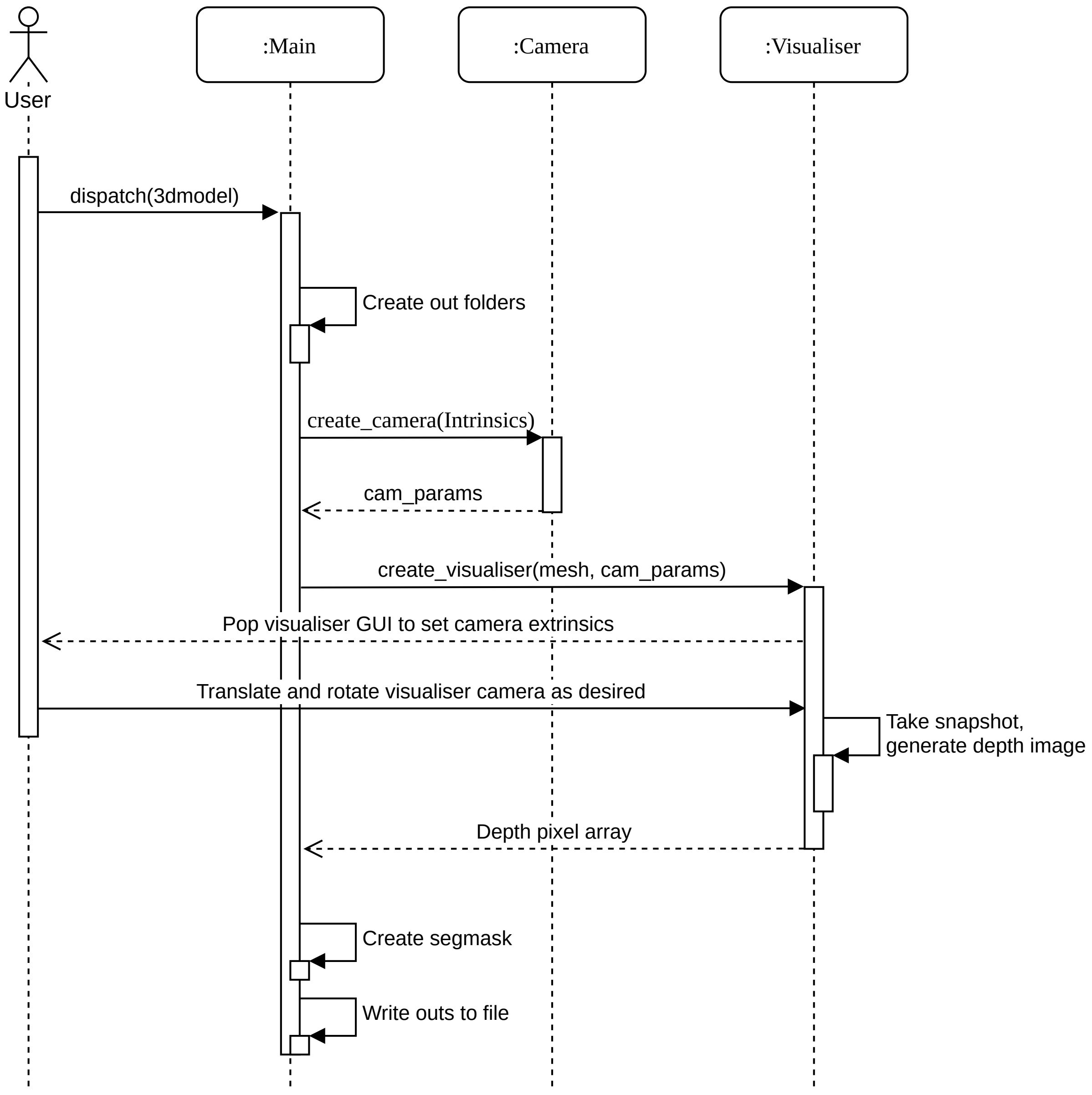

User
:Main
:Camera
:Visualiser
dispatch(3dmodel)
Create out folders
create_camera(Intrinsics)
cam_params
create_visualiser(mesh, cam_params)
Pop visualiser GUI to set camera extrinsics
Translate and rotate visualiser camera as desired
Take snapshot, generate depth image
Depth pixel array
Create segmask
Write outs to file

Figure A.4: UML Sequence Diagram for 3D to depth map utility.

B Grasp Methods Sensitivity Analysis Visualisations

All the visualisations of the qualitative analysis performed in section 6.1 are shown in Figure B.1 to Figure B.32.

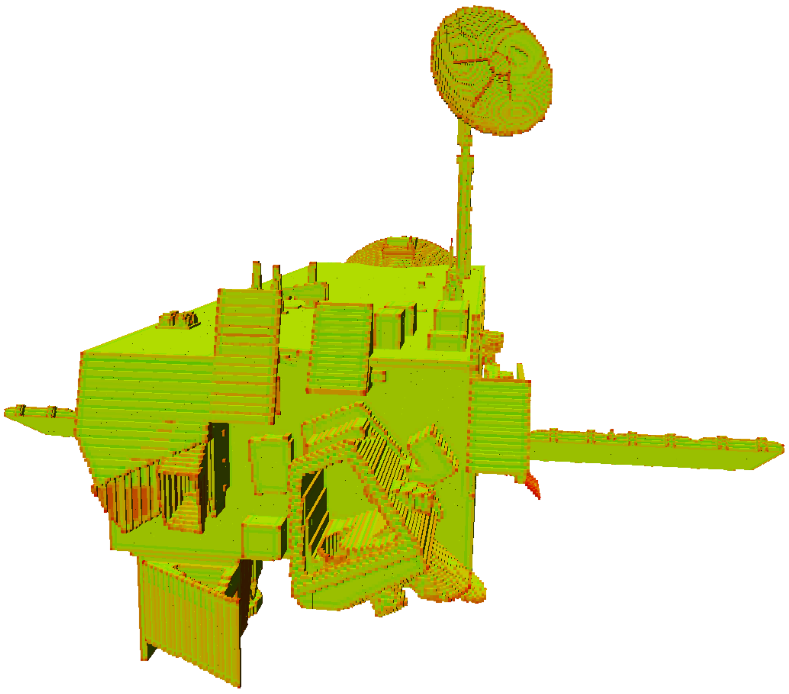


Figure B.1: R61 Mk0 viewpoint 1.

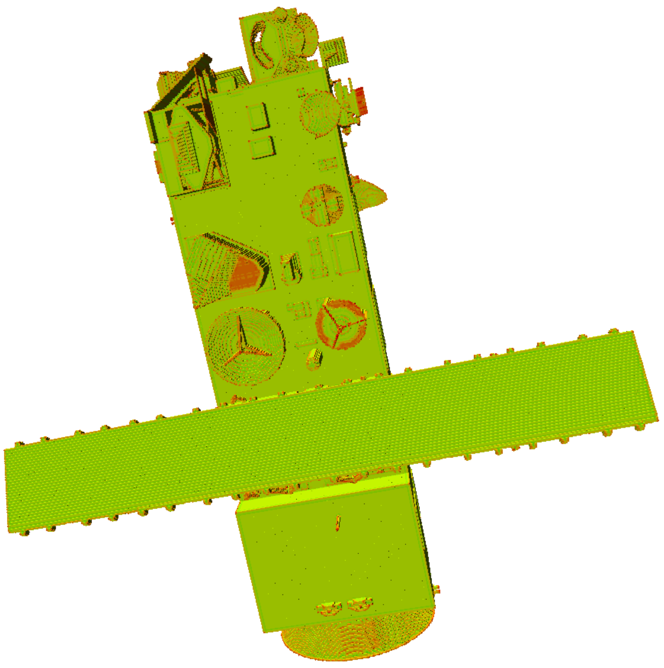


Figure B.2: R61 Mk0 viewpoint 2.

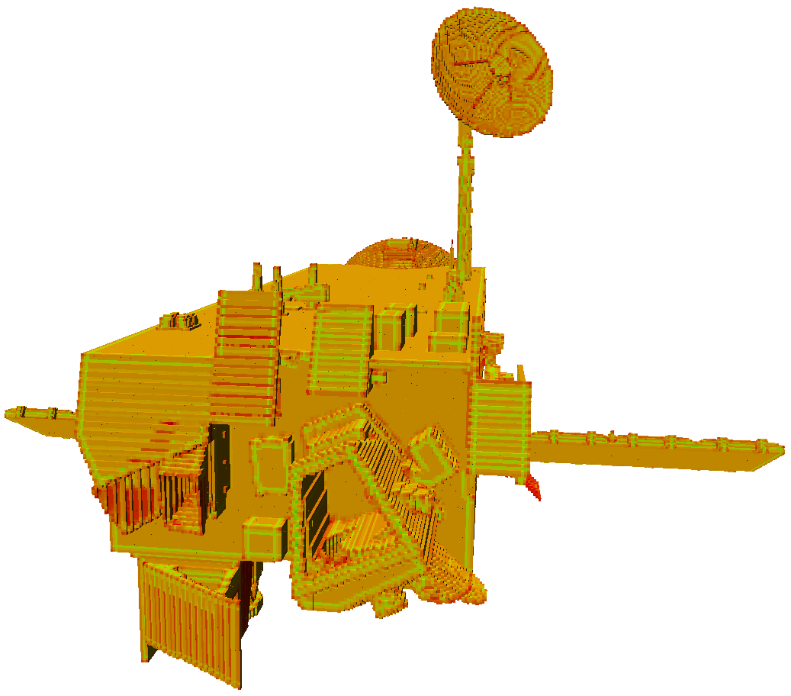


Figure B.3: R61 Mk1 viewpoint 1.

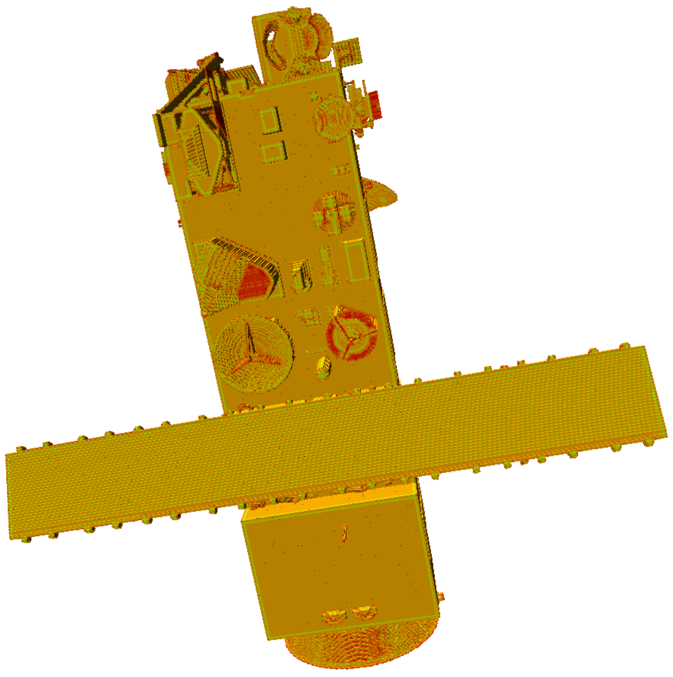


Figure B.4: R61 Mk1 viewpoint 2.

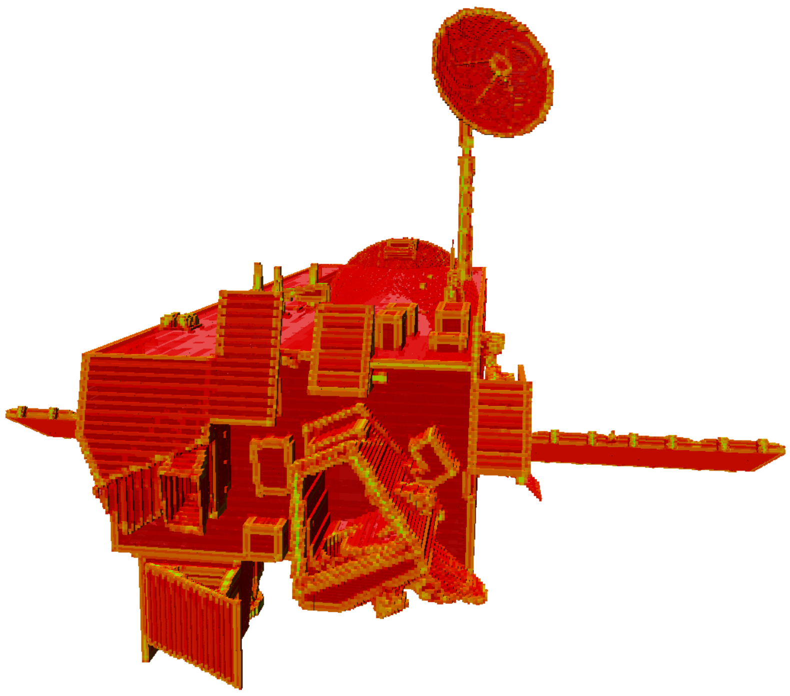


Figure B.5: R61 Mk2 viewpoint 1.

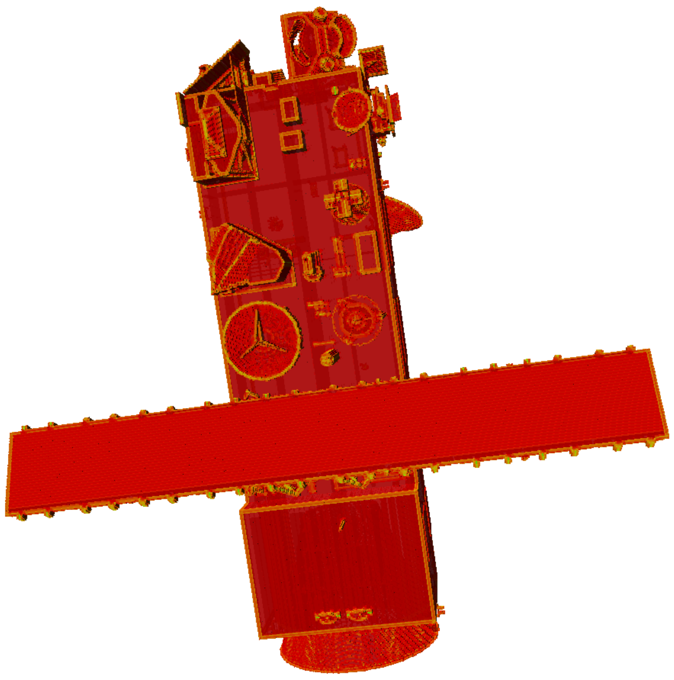


Figure B.6: R61 Mk2 viewpoint 2.

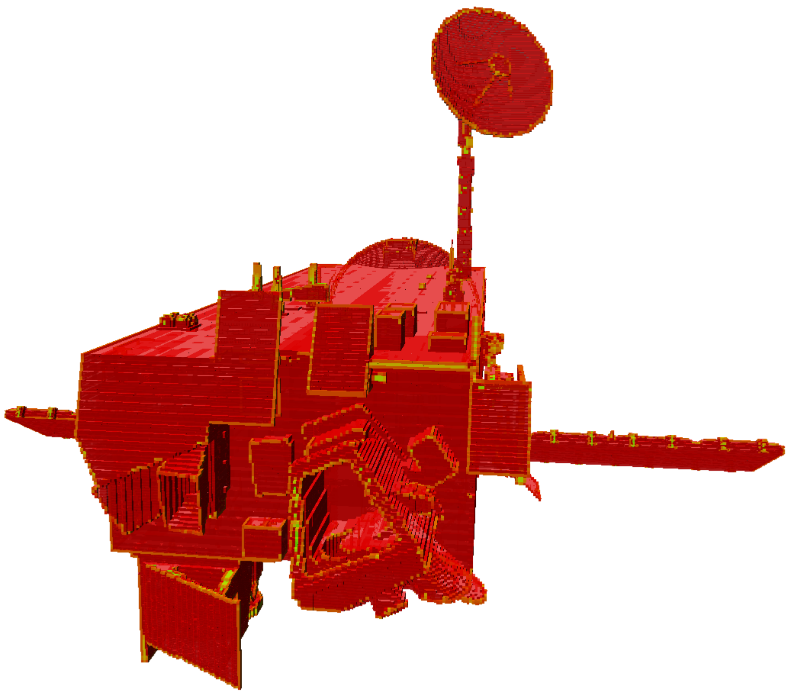


Figure B.7: R61 Mk3 viewpoint 1.

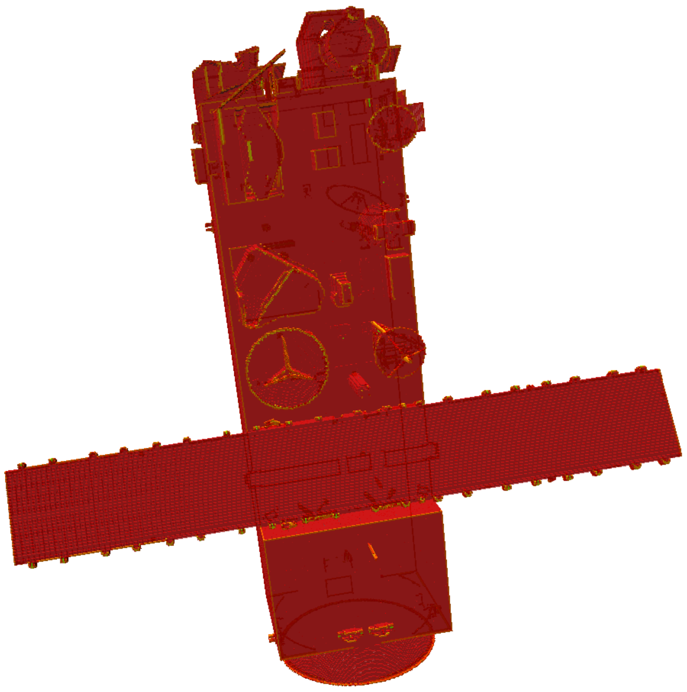


Figure B.8: R61 Mk3 viewpoint 2.

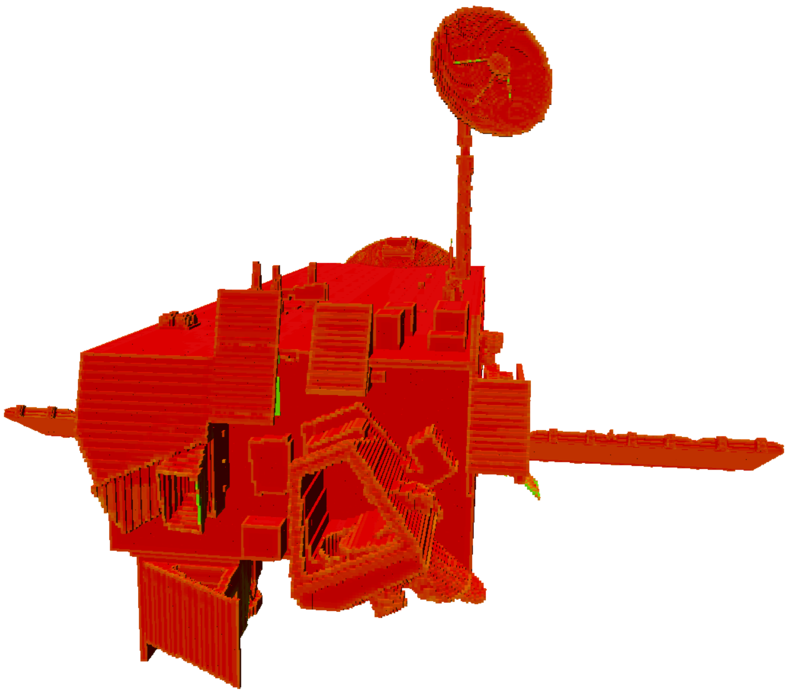


Figure B.9: R82 Mk0 viewpoint 1.

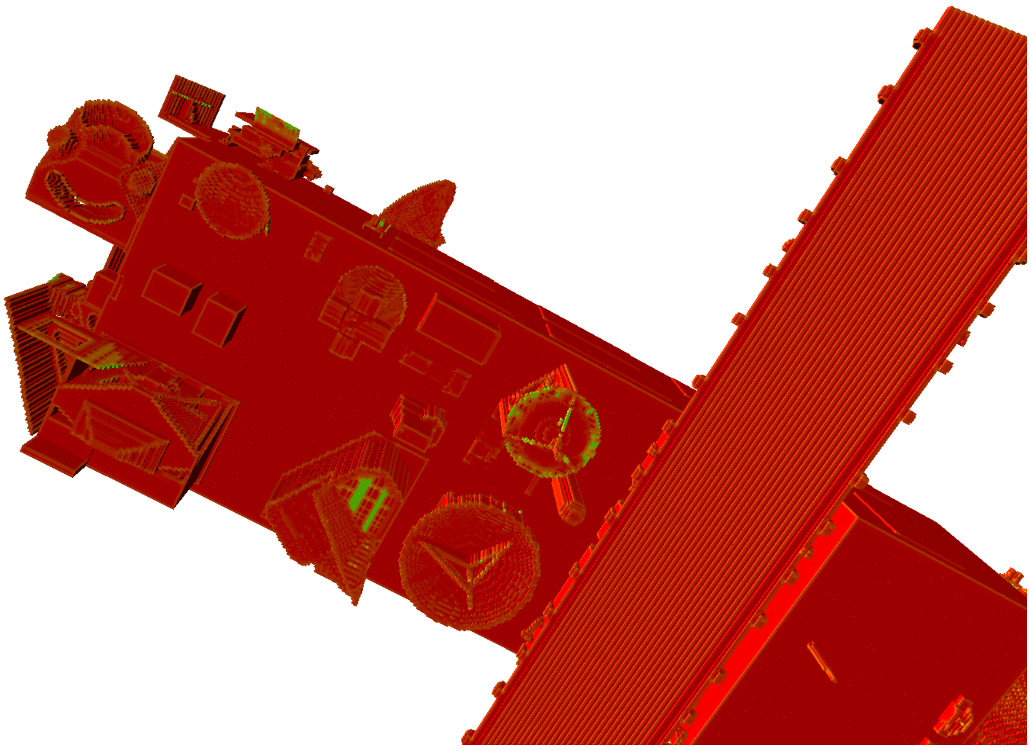


Figure B.10: R82 Mk0 viewpoint 2.

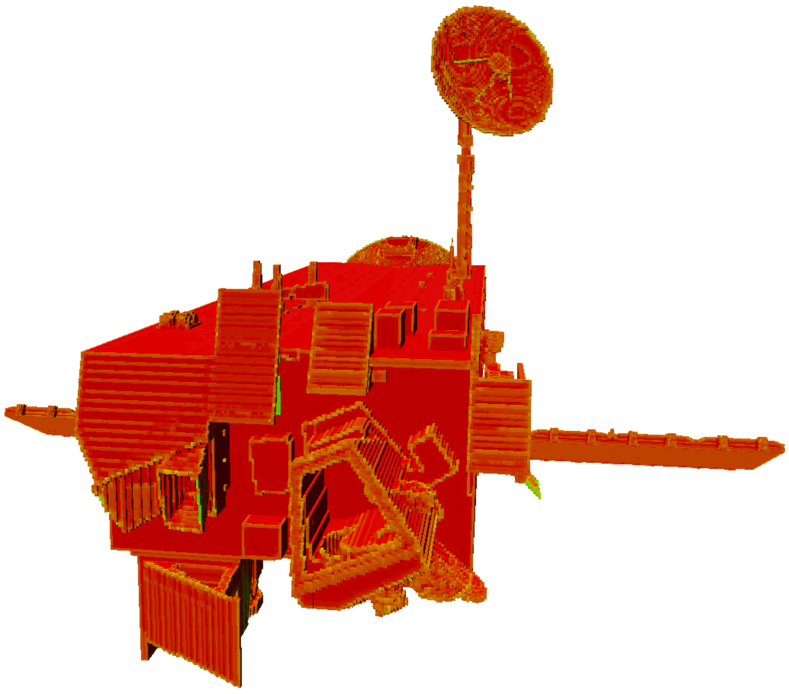


Figure B.11: R82 Mk1 viewpoint 1.

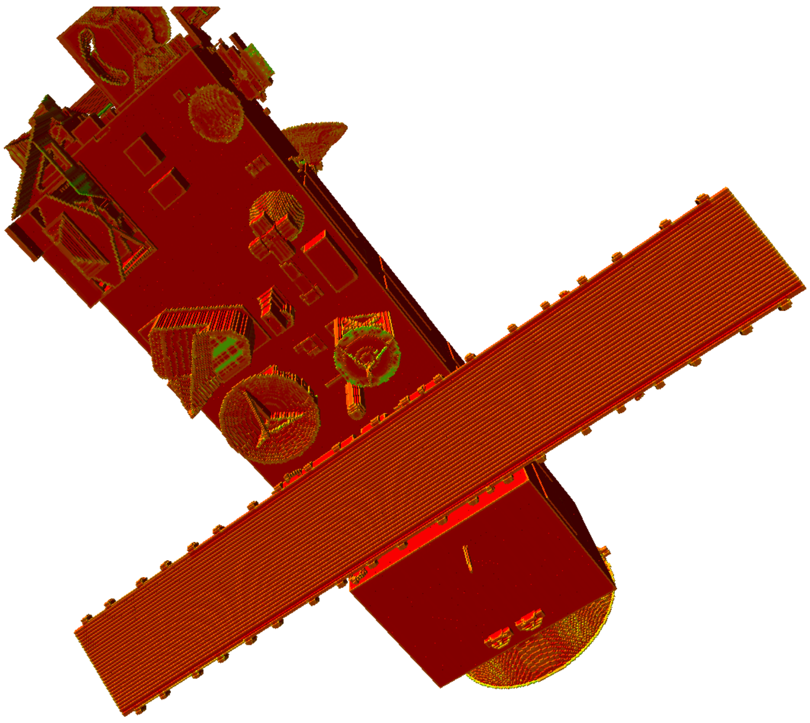


Figure B.12: R82 Mk1 viewpoint 2.

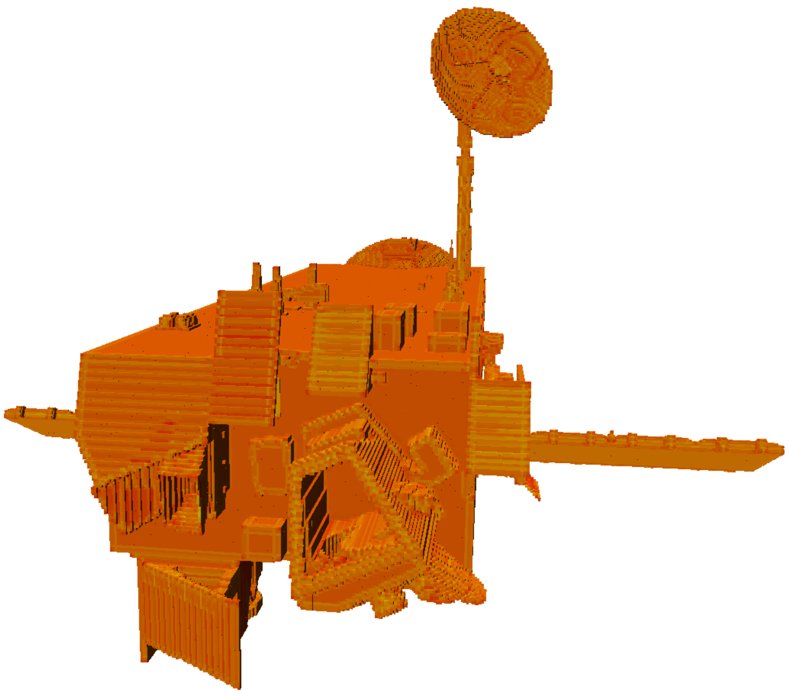


Figure B.13: R83 Mk0 viewpoint 1.

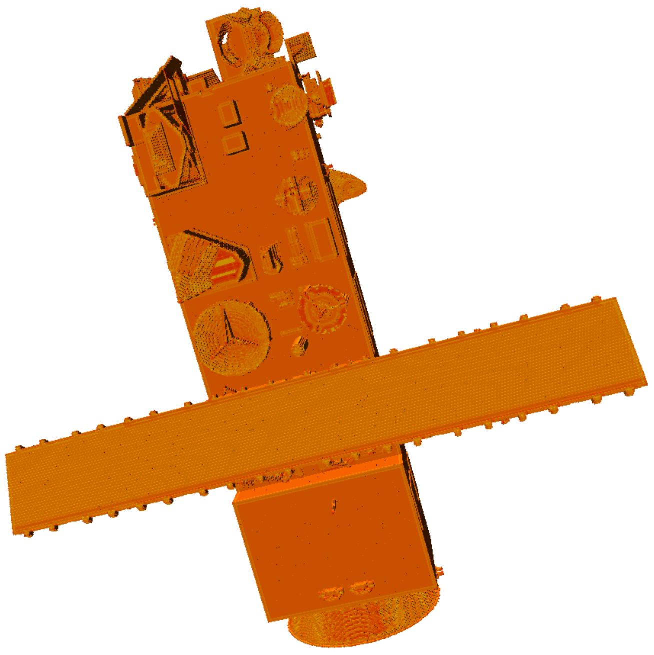


Figure B.14: R83 Mk0 viewpoint 2.

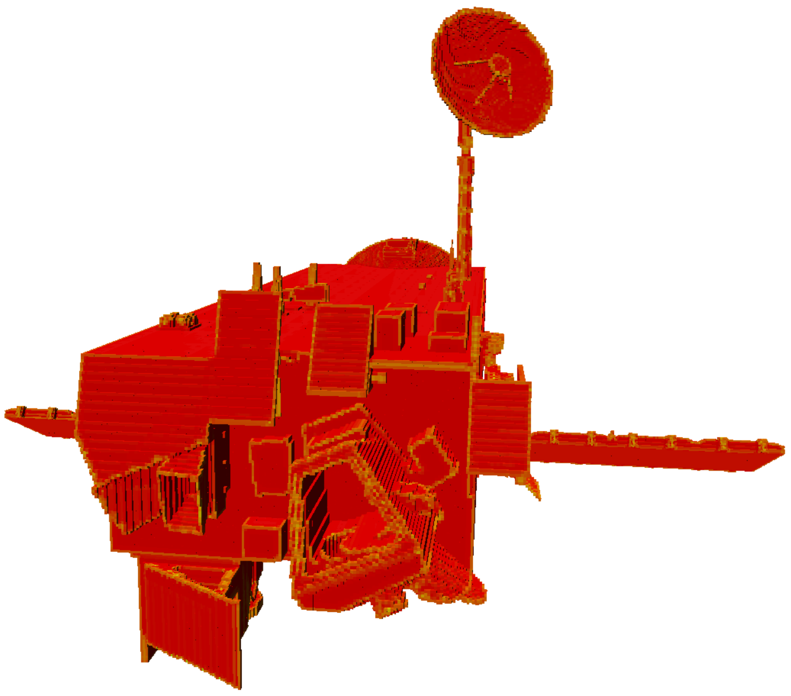


Figure B.15: R83 Mk1 viewpoint 1.

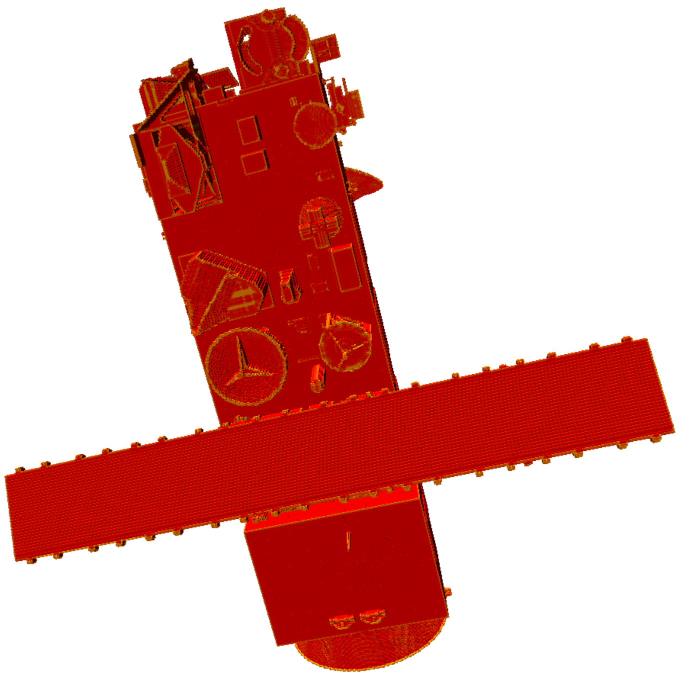


Figure B.16: R83 Mk1 viewpoint 2.

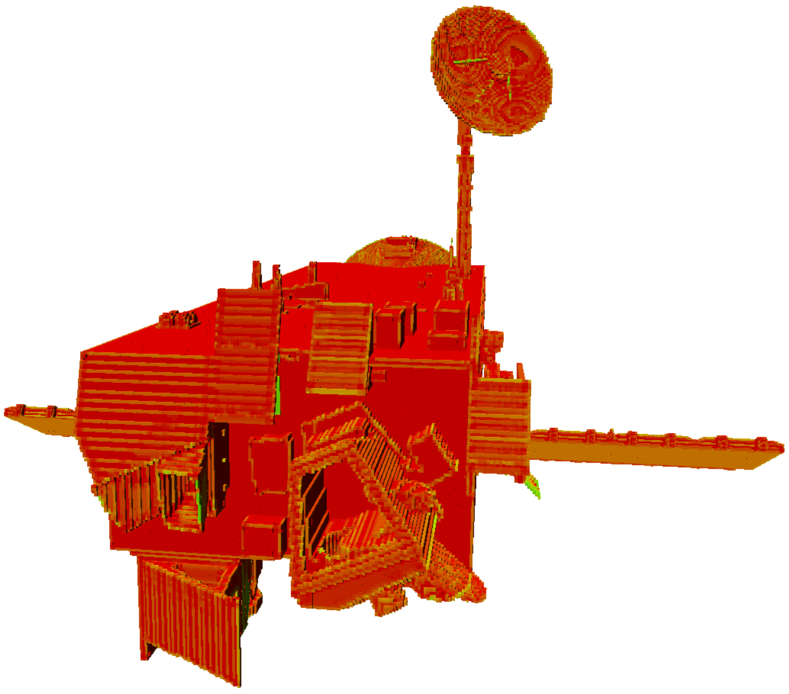


Figure B.17: R84 Mk0 viewpoint 1.

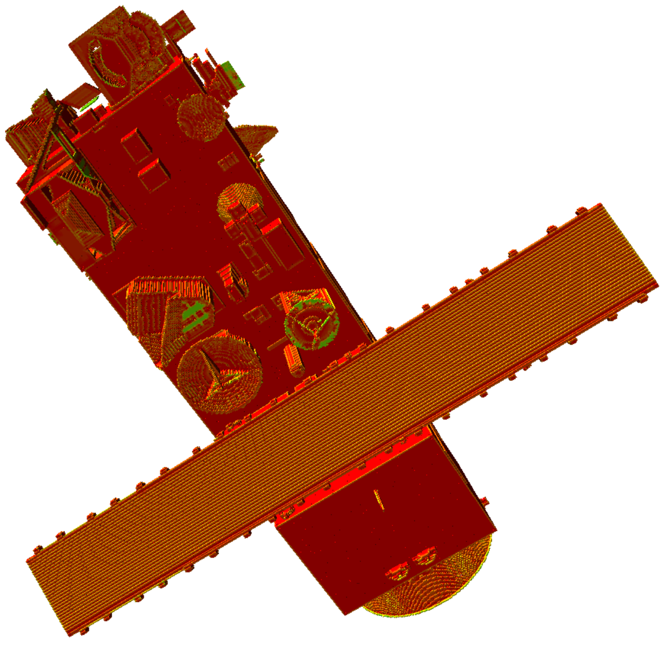


Figure B.18: R84 Mk0 viewpoint 2.

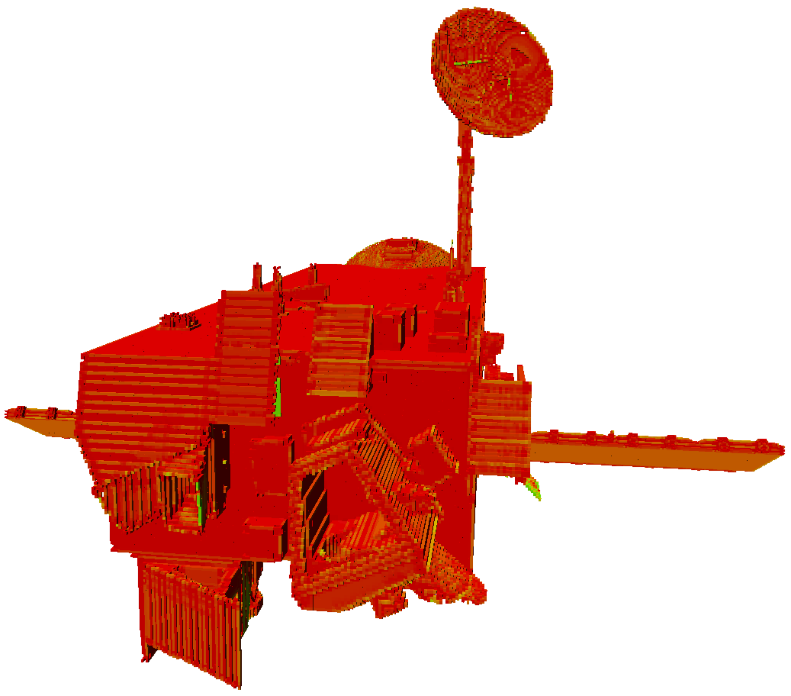


Figure B.19: R84 Mk1 viewpoint 1.

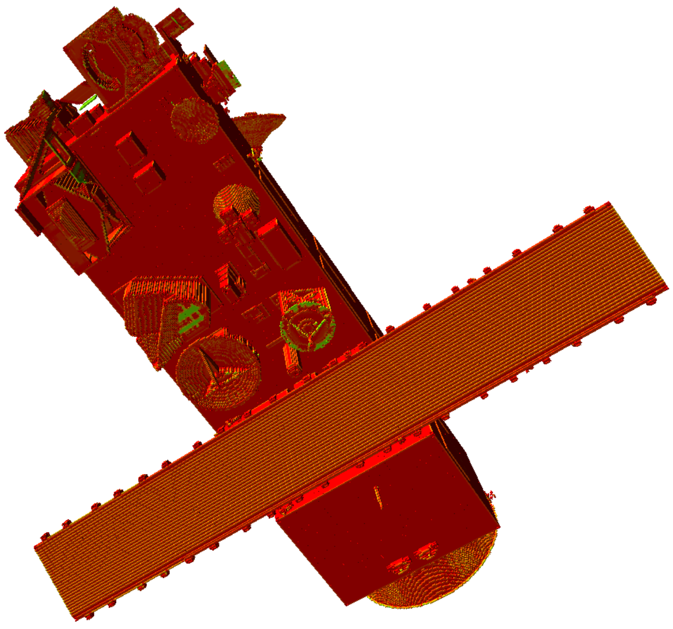


Figure B.20: R84 Mk1 viewpoint 2.

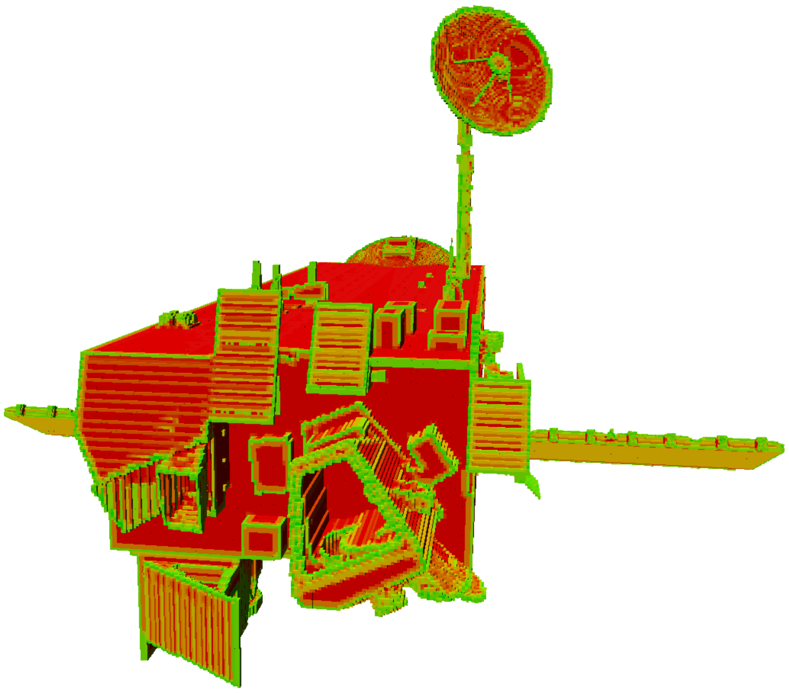


Figure B.21: R85 Mk0 viewpoint 1.

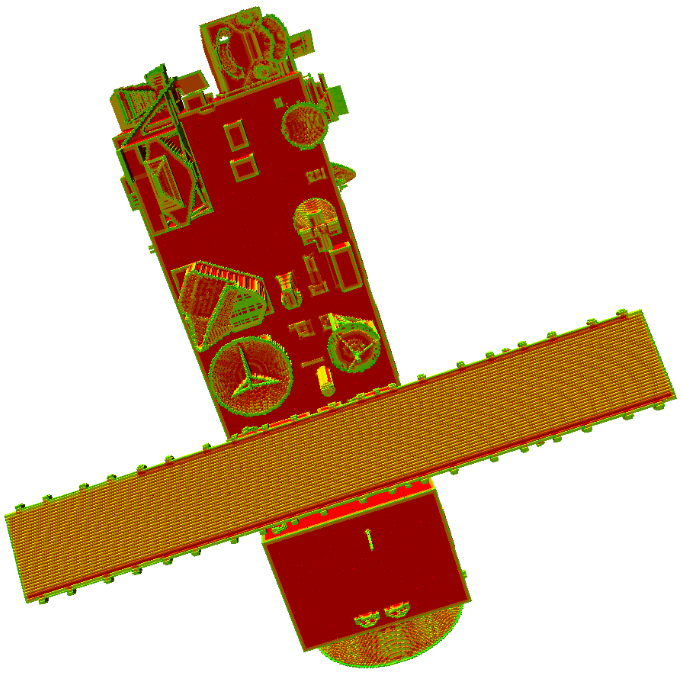


Figure B.22: R85 Mk0 viewpoint 2.

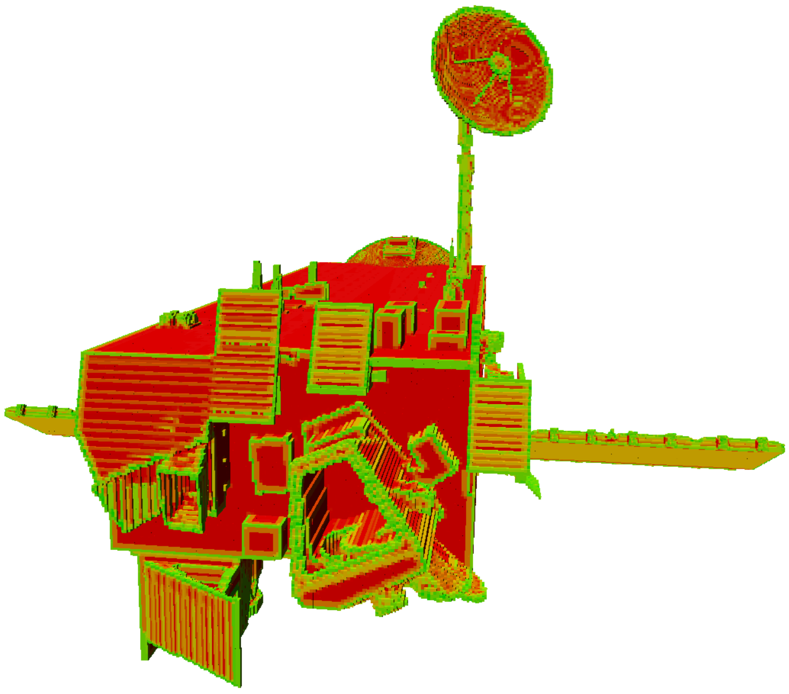


Figure B.23: R85 Mk1 viewpoint 1.

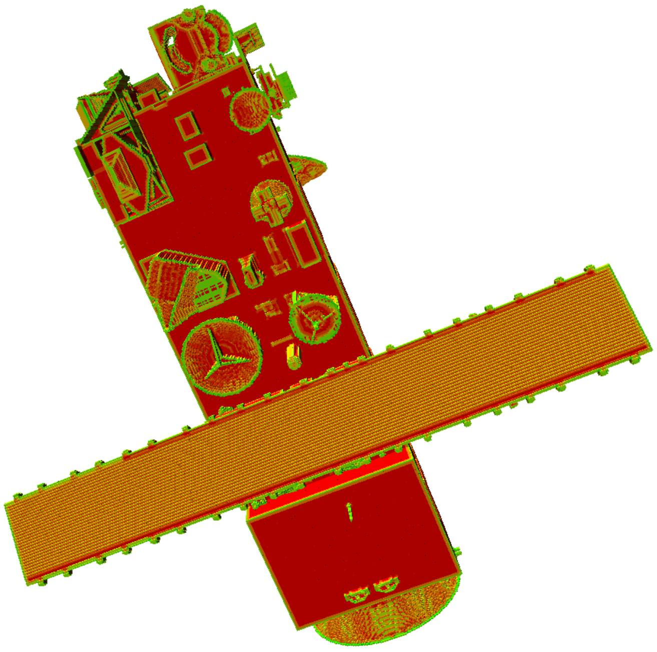


Figure B.24: R85 Mk1 viewpoint 2.

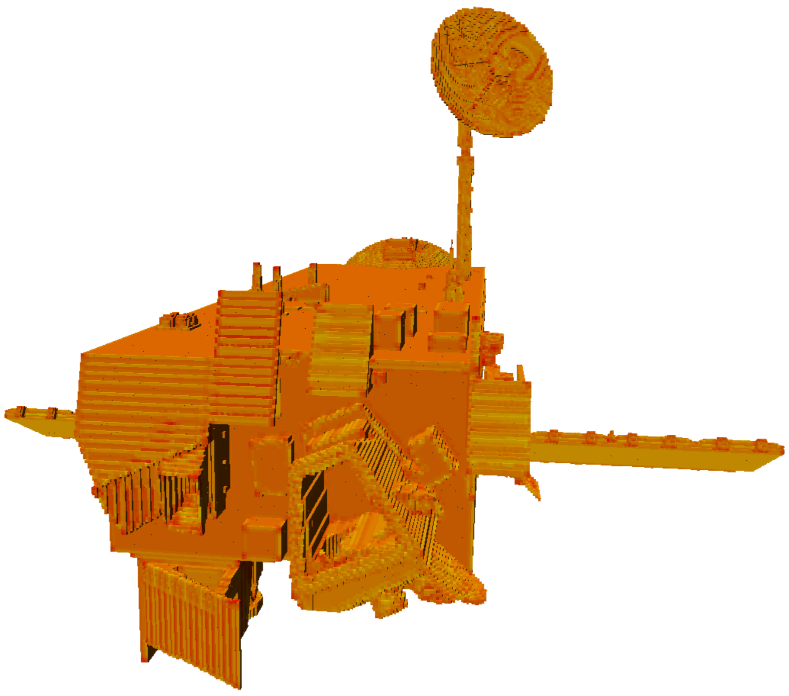


Figure B.25: R86 Mk0 viewpoint 1.

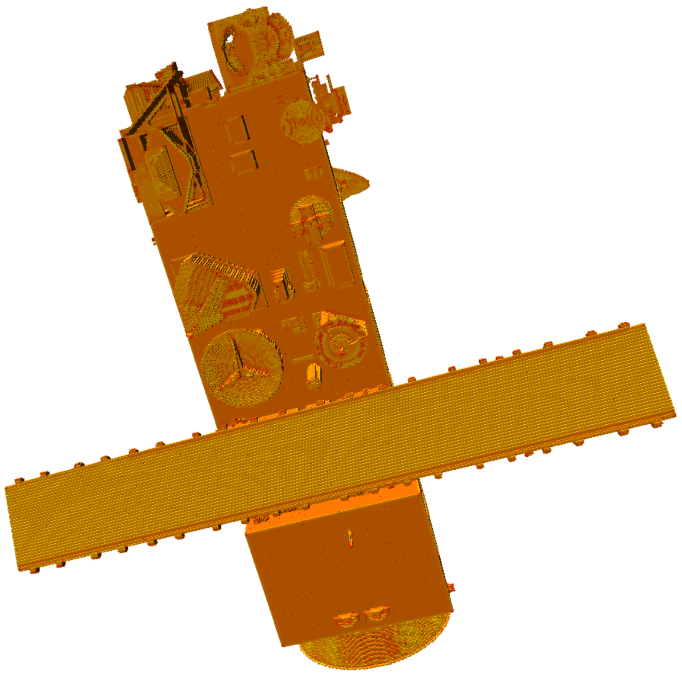


Figure B.26: R86 Mk0 viewpoint 2.

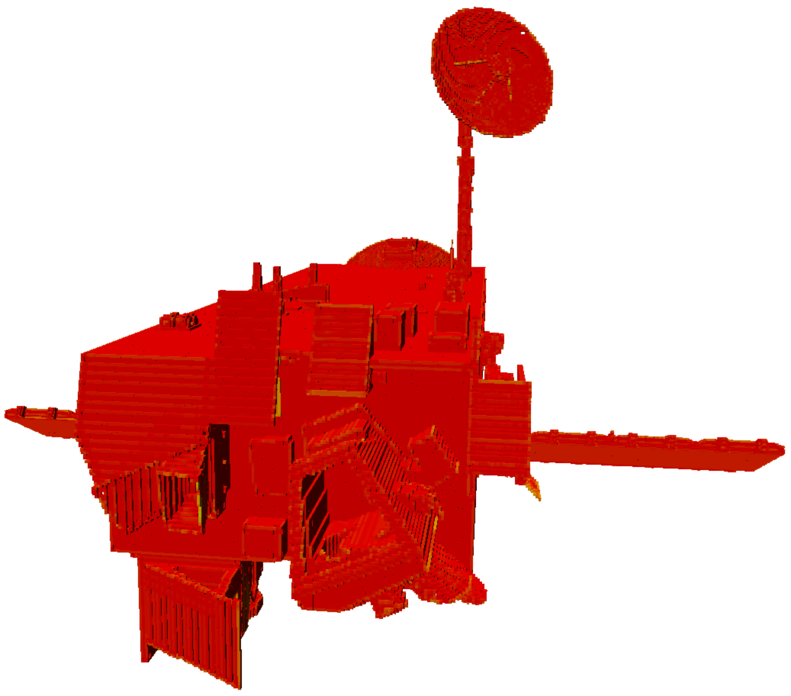


Figure B.27: R86 Mk1 viewpoint 1.

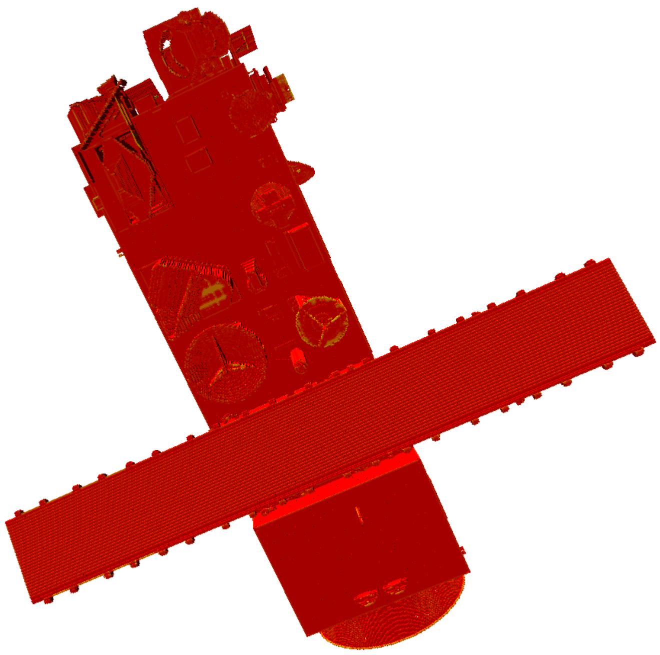


Figure B.28: R86 Mk1 viewpoint 2.

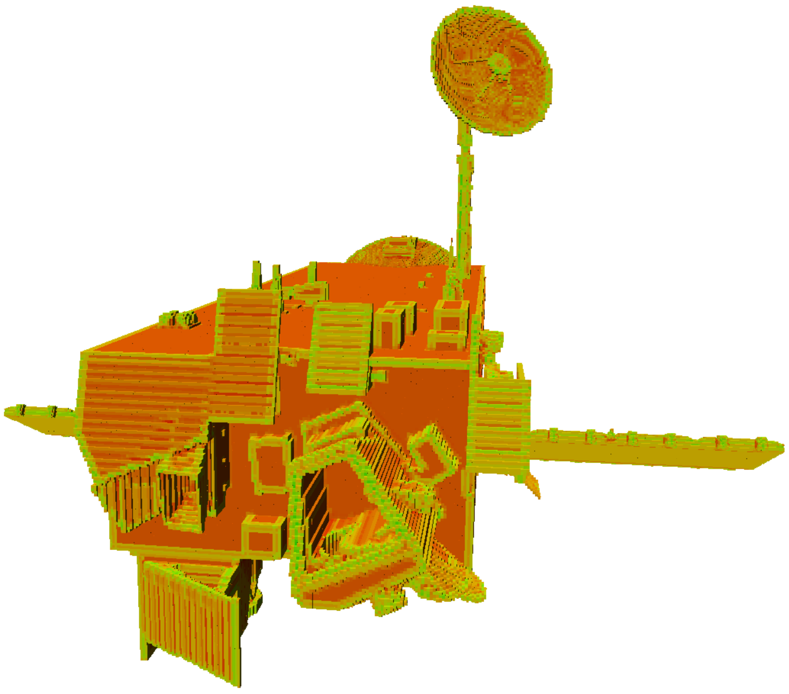


Figure B.29: R87 Mk0 viewpoint 1.

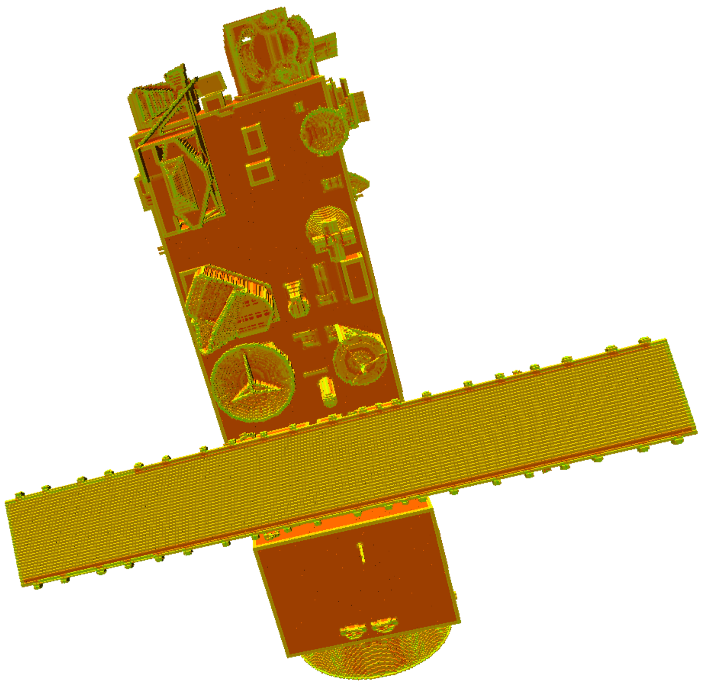


Figure B.30: R87 Mk0 viewpoint 2.

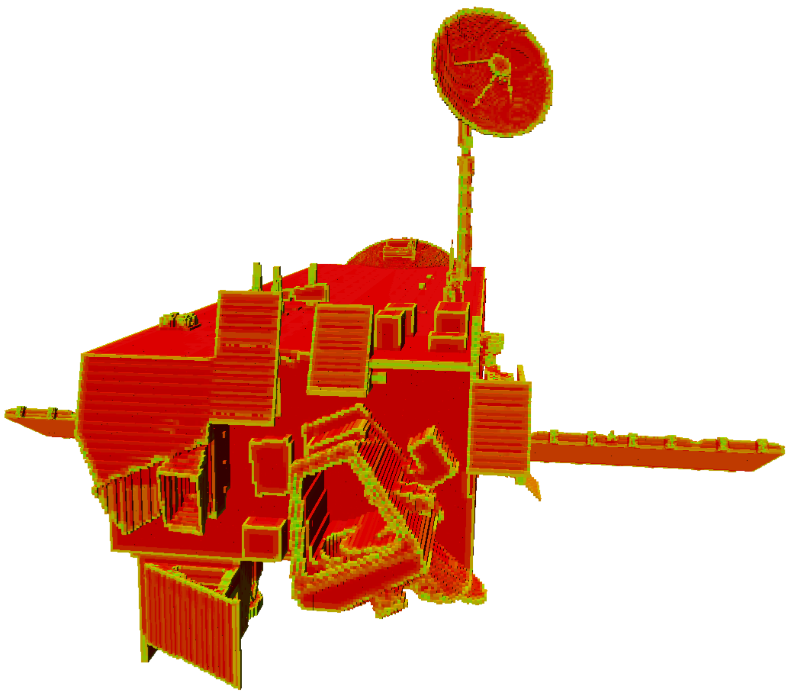


Figure B.31: R87 Mk1 viewpoint 1.

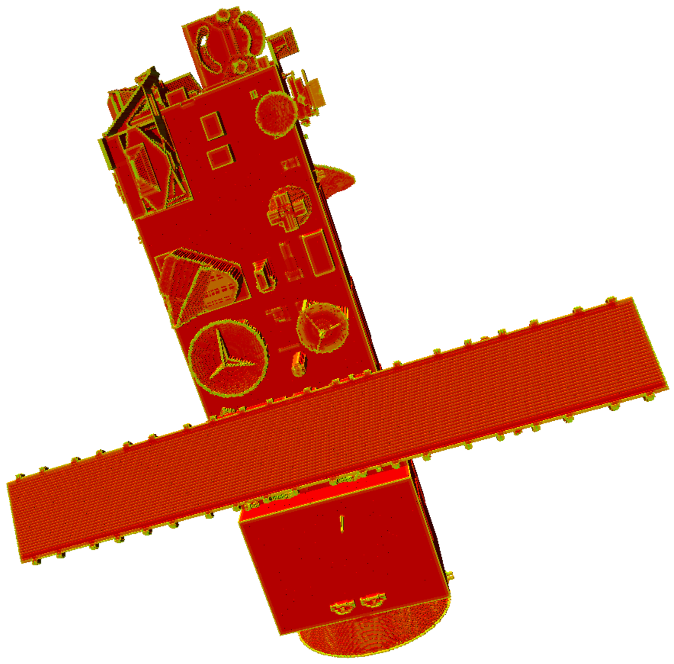


Figure B.32: R87 Mk1 viewpoint 2.

Bibliography

[1] J. Dumoulin. “Project gemini,” NASA. (2000), [Online]. Available: `https://science.ksc.nasa.gov/history/gemini/gemini.html` (visited on 06/17/2020).

[2] W. Fehse, *Automated rendezvous and docking of spacecraft*. Cambridge University Press, 2003, vol. 16.

[3] D. M. Harland, *The story of space station MIR*. Springer Science & Business Media, 2007.

[4] P. Baker, *The Story of Manned Space Stations: An Introduction*. Springer Science & Business Media, 2007.

[5] Northrop Grumman Corporation, *Spacelogistics/mev-2 multimedia files*.

[6] D. S. Portree, *Mir hardware heritage*. Lyndon B. Johnson Space Center, 1995, vol. 1357.

[7] R. B. Friend, “Orbital express program summary and mission overview,” in *Proceedings of SPIE, the International Society for Optical Engineering*, vol. 6958, 2008, p. 695 803.

[8] Northrop Grumman Corporation, “Northrop grumman successfully completes historic first docking of mission extension vehicle with intelsat 901 satellite,” *Northrop Grumman Newsroom*, Feb. 26, 2020.

[9] G. Jorgensen and E. Bains, “Srms history, evolution and lessons learned,” in *AIAA SPACE 2011 Conference & Exposition*, 2011.

[10] A. Ogilvie, J. Allport, M. Hannah, and J. Lymer, “Autonomous satellite servicing using the orbital express demonstration manipulator system,” in *Proc. of the 9th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS’08)*, 2008, pp. 25–29.

[11] L. Tarabini, J. Gil, F. Gandia, M. Á. Molina, J. M. D. Cura, and G. Ortega, “Ground guided cx-olev rendez-vous with uncooperative geostationary satellite,” in *57th International Astronautical Congress*, vol. 61, 2006, pp. 312–325.

[12] C. Kaiser, F. Sjöberg, J. M. Delcura, and B. Eilertsen, “Smart-olev—an orbital life extension vehicle for servicing commercial spacecrafts in geo,” *Acta Astronautica*, vol. 63, no. 1-4, pp. 400–410, 2008.

[13] M. Richard. “Cleanspace one,” EPFL. (2019), [Online]. Available: `https://www.epfl.ch/research/domains/epfl-space-center/spaceresearch/cleanspaceone_1/` (visited on 06/25/2020).

[14] M. Richard, L. G. Kronig, F. Belloni, *et al.*, “Uncooperative rendezvous and docking for microsats,” in *6th International Conference on Recent Advances in Space Technologies, RAST 2013,*, 2013.

[15] I. T. Mitchell. “Draper laboratory overview of rendezvous and capture operations,” Draper Laboratory. (2010), [Online]. Available: `https://nexis.gsfc.nasa.gov/workshop_2010/day3/Ian_Mitchell/Rendezvous_and_ProxOps_IMitchell.pdf` (visited on 06/29/2020).

[16] E. Bains, C. Price, and L. Walter, “Track and capture of the orbiter with the space station remote manipulator system,” Tech. Rep., 1987.

[17] G. Pascoe, W. Maddern, M. Tanner, P. Piniés, and P. Newman, “Nid-slam: Robust monocular slam using normalised information distance,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1435–1444.

[18] W. De Jongh, H. Jordaan, and C. Van Daalen, “Experiment for pose estimation of uncooperative space debris using stereo vision,” *Acta Astronautica*, vol. 168, pp. 164–173, 2020.

[19] C. Bergin. “Asc’s 3d flash lidar camera selected for osiris-rex asteroid mission,” NASASpaceflight.com. (2012), [Online]. Available: `https://www.nasaspaceflight.com/2012/05/ascs-lidar-camera-osiris-rex-asteroid-mission/` (visited on 07/02/2020).

[20] C. S. Bamji, S. Mehta, B. Thompson, *et al.*, “Impixel 65nm bsi 320mhz demodulated tof image sensor with 3µm global shutter pixels and analog binning,” in *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*, IEEE, 2018, pp. 94–96.

[21] K. Shahid and G. Okouneva, “Intelligent lidar scanning region selection for satellite pose estimation,” *Computer Vision and Image Understanding*, vol. 107, no. 3, pp. 203–209, 2007.

[22] R. Pinson, R. Howard, and A. Heaton, “Orbital express advanced video guidance sensor: Ground testing, flight results and comparisons,” in *AIAA guidance, navigation and control conference and exhibit*, 2008, p. 7318.

[23] L. Blarre, N. Perrimon, C. Moussu, P. Da Cunha, and S. Strandmoe, “Atv videometer qualification,” in *55th International Astronautical Congress of the International Astronautical Federation, the International Academy of Astronautics, and the International Institute of Space Law*, 2004, A–3.

[24] Y. Xu, S. Tuttas, L. Hoegner, and U. Stilla, “Voxel-based segmentation of 3d point clouds from construction sites using a probabilistic connectivity model,” *Pattern Recognition Letters*, vol. 102, pp. 67–74, 2018.

[25] S. Gebhardt, E. Payzer, L. Salemann, A. Fettinger, E. Rotenberg, and C. Seher, “Polygons, point-clouds and voxels: A comparison of high-fidelity terrain representations,” in *Simulation Interoperability Workshop and Special Workshop on Reuse of Environmental Data for Simulation—Processes, Standards, and Lessons Learned*, 2009, p. 9.

[26] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, “Towards 3d point cloud based object maps for household environments,” *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 927–941, 2008.

[27] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, “Surface reconstruction from unorganized points,” in *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, vol. 26, 1992, pp. 71–78.

[28] N. J. Mitra and A. Nguyen, “Estimating surface normals in noisy point cloud data,” in *Proceedings of the nineteenth annual symposium on Computational geometry*, 2003, pp. 322–328.

[29] W. E. Lorensen and H. E. Cline, “Marching cubes: A high resolution 3d surface construction algorithm,” *ACM siggraph computer graphics*, vol. 21, no. 4, pp. 163–169, 1987.

[30] J. Bohg, A. Morales, T. Asfour, and D. Kragic, “Data-driven grasp synthesis—a survey,” *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014.

[31] V.-D. Nguyen, “Constructing force-closure grasps,” *The International Journal of Robotics Research*, vol. 7, no. 3, pp. 3–16, 1988.

[32] C. Ferrari and J. Canny, “Planning optimal grasps,” in *Proceedings 1992 IEEE International Conference on Robotics and Automation*, 1992, pp. 2290–2295.

[33] I. Lenz, H. Lee, and A. Saxena, “Deep learning for detecting robotic grasps,” *The International Journal of Robotics Research*, vol. 34, no. 4, pp. 705–724, 2015.

[34] A. Miller, S. Knoop, H. Christensen, and P. Allen, “Automatic grasp planning using shape primitives,” in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 2, 2003, pp. 1824–1829.

[35] J. Redmon and A. Angelova, “Real-time grasp detection using convolutional neural networks,” in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1316–1322.

[36] Y. Jiang, S. Moseson, and A. Saxena, “Efficient grasping from rgbd images: Learning using a new rectangle representation,” in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3304–3311.

[37] J. Mahler, J. Liang, S. Niyaz, *et al.*, “Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics,” in *Robotics: Science and Systems 2017*, vol. 13, 2017.

[38] A. Zeng, S. Song, K.-T. Yu, *et al.*, “Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3750–3757.

[39] D. Kraft, N. Pugeault, E. Baseski, *et al.*, “Birth of the object: Detection of objectness and extraction of object shape through object–action complexes,” *International Journal of Humanoid Robotics*, vol. 5, no. 2, pp. 247–265, 2008.

[40] M. Popovic, G. Kootstra, J. A. Jorgensen, D. Kragic, and N. Kruger, “Grasping unknown objects using an early cognitive vision system for general scene understanding,” in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 987–994.

[41] G. Bone, A. Lambert, and M. Edwards, “Automated modeling and robotic grasping of unknown three-dimensional objects,” in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 292–298.

[42] M. Richtsfeld and M. Vincze, “Grasping of unknown objects from a table top,” *Workshop on Vision in Action: Efficient strategies for cognitive agents in complex environments*, 2008.

[43] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel, “Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding,” in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 2308–2315.

[44] K. Hsiao, S. Chitta, M. Ciocarlie, and E. G. Jones, “Contact-reactive grasping of objects with partial shape information,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 1228–1235.

[45] P. Brook, M. Ciocarlie, and K. Hsiao, “Collaborative grasp planning with multiple object representations,” in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2851–2858.

[46] J. Bohg, M. Johnson-Roberson, B. Leon, *et al.*, “Mind the gap - robotic grasping under incomplete observation,” in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 686–693.

[47] J. Stueckler, R. Steffens, D. Holz, and S. Behnke, “Real-time 3d perception and efficient grasp planning for everyday manipulation tasks.,” *Proc. of the European Conf. on Mobile Robots (ECMR)*, pp. 177–182, 2011.

[48] E. Klingbeil, D. Rao, B. Carpenter, V. Ganapathi, A. Y. Ng, and O. Khatib, “Grasping with application to an autonomous checkout robot,” in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2837–2844.

[49] A. Maldonado, U. Klank, and M. Beetz, “Robotic grasping of unmodeled objects using time-of-flight range data and finger torque information,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 2586–2591.

[50] Z.-C. Marton, D. Pangercic, N. Blodow, J. Kleinehellefort, and M. Beetz, “General 3d modelling of novel objects from a single view,” in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 3700–3705.

[51] V. Lippiello, F. Ruggiero, B. Siciliano, and L. Villani, “Visual grasp planning for unknown objects using a multifingered robotic hand,” *IEEE-ASME Transactions on Mechatronics*, vol. 18, no. 3, pp. 1050–1059, 2013.

[52] C. Dune, E. Marchand, C. Collowet, and C. Leroux, “Active rough shape estimation of unknown objects,” in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 3622–3627.

[53] B. Kehoe, D. Berenson, and K. Goldberg, “Toward cloud-based grasping with uncertainty in shape: Estimating lower bounds on achieving force closure with zero-slip push grasps,” in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 576–583.

[54] A. Morales, P. J. Sanz, A. P. del Pobil, and A. H. Fagg, “Vision-based three-finger grasp synthesis constrained by hand geometry,” *Robotics and Autonomous Systems*, vol. 54, no. 6, pp. 496–512, 2006.

[55] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016, ch. 38.

[56] C. Zou, E. Yumer, J. Yang, D. Ceylan, and D. Hoiem, “3d-prnn: Generating shape primitives with recurrent neural networks,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 900–909.

[57] K. I. Laws, “Rapid texture identification,” in *Image Processing for Missile Guidance*, vol. 238, 1980, pp. 376–381.

[58] H. Zhang, W. Gao, X. Chen, and D. Zhao, “Object detection using spatial histogram features,” *Image and Vision Computing*, vol. 24, no. 4, pp. 327–341, 2006.

[59] A. Saxena, J. Driemeyer, and A. Y. Ng, “Robotic grasping of novel objects using vision,” *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 157–173, 2008.

[60] M. J. Hegedus, K. Gupta, and M. Mehrandezh, “Generalized grasping for mechanical grippers for unknown objects with partial point cloud representations.,” *arXiv preprint arXiv:2006.12676*, 2020.

[61] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg, “Dex-net 3.0: Computing robust robot suction grasp targets in point clouds using a new analytic model and deep learning,” *arXiv preprint arXiv:1709.06670*, 2017.

[62] J. Mahler, M. Matl, V. Satish, *et al.*, “Learning ambidextrous robot grasping policies,” *Science Robotics*, vol. 4, no. 26, eaau4984, 2019.

[63] C. Eppner, A. Mousavian, and D. Fox, “A billion ways to grasp: An evaluation of grasp sampling schemes on a dense, physics-based grasp data set,” *arXiv preprint arXiv:1912.05604*, 2019.

[64] M. Schwendener, M. Vilella, and P. Bänninger, *Grals operations manual*, European Space Agency, Mar. 2020.

[65] ——, *Grals design and configuration document*, European Space Agency, Mar. 2020.

[66] KUKA Robotics Corporation, *Kr agilus sixx wp with w and c variants, operating instructions*.

[67] J. R. Sánchez-Ibánez, C. J. Pérez-del-Pulgar, M. Azkarate, L. Gerdes, and A. García-Cerezo, “Dynamic path planning for reconfigurable rovers using a multi-layered grid,” *Engineering Applications of Artificial Intelligence*, vol. 86, pp. 32–42, 2019.

[68] S. Rusinkiewicz and M. Levoy, “Efficient variants of the icp algorithm,” in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, 2001, pp. 145–152. DOI: `10.1109/IM.2001.924423`.

[69] Q.-Y. Zhou, J. Park, and V. Koltun, “Open3D: A modern library for 3D data processing,” *arXiv:1801.09847*, 2018.

[70] A. Gini, *Don kessler on envisat and the kessler syndrome*, 2012.

[71] P. Min, *Binvox*, `http://www.patrickmin.com/binvox`, 2004 - 2019.

[72] F. S. Nooruddin and G. Turk, “Simplification and repair of polygonal models using volumetric techniques,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 2, pp. 191–205, 2003.