UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Planning and Perception for Robotic Manipulators in Semi-structured Environments

by Fouad Sukkar

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of Prof. Robert Fitch

University of Technology Sydney
Faculty of Engineering and Information Technology

February 2022

Certificate of Original Authorship

I, Fouad Sukkar declare that this thesis, is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Mechanical and Mechatronic Engineering at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Production Note:
Signature removed prior to publication.

Fouad Sukkar

February 2022

Planning and Perception for Robotic Manipulators in Semi-structured Environments

by

Fouad Sukkar

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

Abstract

New applications of robotic manipulators require a far greater degree of autonomy than traditionally has been needed. Industrial robot manipulators such as collaborative robots are designed for advanced manufacturing applications where they should operate safely in dynamic work environments shared with humans and should be able to adapt quickly to perform a variety of tasks. Manipulators for fruit harvesting in agricultural robotics need to operate quickly in semi-structured outdoor environments in order to perform at a commercially viable level.

These applications require many components to work together robustly and efficiently in order to be viable for real world use. For example, robotic harvesting involves task and motion planning, task space control, active perception, object segmentation and grasping. Extending this to multiple robots requires coordination among agents which is non-trivial and further increases problem complexity. Existing methods for solving such problems in real-time often resort to making coarse approximations to one or many parts of the problem, for example a simplified motion or sensor model. This can lead to undesirable behaviour particularly for highly non-linear systems such as robotic arms or complex sensor models such as those for RGB-D cameras. Furthermore, when searching for objects it remains a challenge to effectively balance between exploration of a scene and targeted observation of key regions.

This thesis is concerned with developing component algorithms that solve such problems effectively and exhibit computational behaviour that is well-understood analytically. Key to achieving this is the exploitation of known a priori structure in the problem, typical of semi-structured environments. First, a novel planning framework called Hausdorff approximation planner (HAP) is described which utilises prior knowledge for building a library of trajectories which are guaranteed to be of bounded length and approximately correspond to geodesics in the task space. These trajectories can be concatenated reliably over long task horizons, adapted quickly online and queried for accurate costs in constant time. Then, a novel targeted information gathering algorithm is presented which is integrated with HAP and other components into a non-myopic active perception framework suitable for systems with complex motion and sensor models and effectively balances between exploration and targeted observations of the scene. This is then generalised into a decentralised coordination framework suitable for flexible multi-robot perception problems that facilitate downstream processes, such as manipulation, object segmentation, and collaborative tasks. To demonstrate the flexibility and effectiveness of these frameworks they are evaluated extensively in various problem settings with several different robotic platforms both in simulation and hardware, including robotic pruning, robotic harvesting, and searching for objects in clutter using multiple robotic manipulators with eye-in-hand sensors.

Acknowledgements

This thesis would not have been possible without the help and support of the many people around me and whom I have met throughout my studies, both locally and abroad. First and foremost, thank you to Prof. Robert Fitch. You have put up with me as a supervisor for a majority of my academic career, spanning multiple degrees, and have never failed me in your advice and guidance. Your ability to communicate technical ideas in a clear and inspiring way never ceases to amaze me and I have learnt to become a better communicator as a consequence. Thank you to my co-supervisors Dr Alen Alempijevic and Dr Chanyeol Yoo for your words of encouragement and invaluable advice.

I am deeply grateful to my fellow researchers from the robotics group at UTS, especially Felix Kong, Giovanni D'urso, Brian Lee, Cadmus To, James Lee, Jennifer Wakulicz and Jayant Khatkar; you have all inspired creativity and helped make my PHD an unforgettable experience. I am thankful for the opportunity to have collaborated and spent time with amazing groups from Oregon State University and Washington State University - Graeme Best, Joseph R. Davidson, Alexander You and Manoj Karkee - my thesis has greatly benefited from your help and advice. To the reading groups I've attended over the past years at UTS and Usyd, I appreciate the invaluable discussions and feedback which helped me refine the ideas in this thesis and produce high quality publications.

Last but not least, thanks to my Mum, Dad, sisters Jessica and Clarissa, and friends and family for your unending support. It has been a challenging time and you have helped me through the best and worst, this thesis is dedicated to you.

Contents

Declaration of Authorship iii

Abstract v

Acknowledgements vii

Contents ix

List of Figures xv

List of Tables xv

List of Algorithms xvii

List of Theorems and Definitions xxi

Nomenclature xxi

1 Introduction 1
- 1.1 Planning In Semi-Structured Environments . . . 3
- 1.2 Manipulator Planning for Perception . . . 5
- 1.3 Limitations of Current Manipulator Planners . . . 6
- 1.4 Planning with Hausdorff Approximations (HAs) . . . 8
- 1.5 Scope . . . 11
  - 1.5.1 Manipulator Task and Motion Planning . . . 11
  - 1.5.2 Information Gathering and Active Perception . . . 12
  - 1.5.3 Task-Space Control and Manipulation . . . 12
  - 1.5.4 Multi-Robot Active Perception . . . 13
  - 1.5.5 Problem Settings . . . 14
- 1.6 Principal Contributions . . . 17
- 1.7 Thesis Outline . . . 19

2 Related Work 21
- 2.1 Planning with Domain Knowledge . . . 21

2.2 General Motion Planners . . . 24
2.2.1 Grid-Based Planning . . . 24
2.2.2 Sampling-Based Planning . . . 25
2.2.3 Trajectory Optimisation . . . 28
2.3 Task and Motion Planning . . . 31
2.4 Coverage Planning . . . 32
2.5 Next-Best-View Planning . . . 32
2.6 Information Gathering . . . 35
2.7 Multi-Robot Active Perception . . . 36

3 Hausdorff Approximation Planner (HAP) 37
3.1 Overview . . . 37
3.1.1 Chapter Outline . . . 40
3.2 Problem Formulation . . . 40
3.2.1 Notation . . . 40
3.2.2 Task and Motion Planning Problem . . . 40
3.2.3 Problem Reduction . . . 41
3.3 Online Motion Planning with HAs . . . 43
3.3.1 Retrieval of Trajectories . . . 43
3.3.2 Modifying Trajectories to Satisfy Tasks . . . 44
3.3.3 Task Sequencing . . . 45
3.3.4 Online Adaptation of Trajectories . . . 45
3.4 Trajectory Library Construction . . . 46
3.4.1 Algorithm Overview . . . 46
3.4.2 Modified Dijkstra Algorithm for Finding HAs . . . 48
3.4.3 Practical Considerations . . . 49
3.5 Motivation for Multiple HAs . . . 55
3.6 Analysis . . . 58
3.6.1 Complexity Analysis . . . 58
3.6.2 Bounded Suboptimality Analysis . . . 59
3.7 HAP Evaluation . . . 62
3.7.1 Experimental Setup . . . 62
3.7.2 Implementation Details . . . 65
3.7.3 Bookshelf Environment with 6D Pose Tasks . . . 66
3.7.4 Task-Space Control . . . 70
3.8 Robotic Pruning . . . 76
3.8.1 Experimental Setup . . . 77
3.8.2 Hardware Results . . . 81
3.9 Summary . . . 82

4 Targeted Information Gathering 83
4.1 Overview . . . 83
4.1.1 Chapter Outline . . . 85
4.2 Problem Formulation . . . 85

4.2.1 Robot and Environment . . . 85
4.2.2 Observation Model . . . 86
4.3 Problem Statement . . . 87
4.4 Viewpoint Evaluation . . . 88
4.4.1 Surface Reconstruction . . . 88
4.4.2 Volumetric Information Formulation . . . 88
4.5 Active Perception Framework . . . 91
4.5.1 Monte Carlo Tree Search . . . 92
4.5.2 Reward Function . . . 93
4.5.3 Computing Realistic Trajectory Costs for Budgeting . . . 95
4.6 Apple Harvesting . . . 96
4.6.1 Experimental Setup . . . 96
4.6.2 Implementation Details . . . 98
4.6.3 Hardware Results . . . 102
4.7 Summary . . . 110

5 Decentralised Coordination Framework 111
5.1 Overview . . . 111
5.1.1 Chapter Outline . . . 112
5.2 Robot Team . . . 112
5.3 Problem Statement . . . 113
5.4 Decentralised Coordinated Planning . . . 113
5.4.1 Decentralised Monte Carlo Tree Search . . . 114
5.4.2 Integration with Active Perception Framework . . . 115
5.4.3 System Execution . . . 116
5.4.4 Problem Complexity Analysis . . . 117
5.5 Searching for Objects with Multiple Robot Manipulators . . . 119
5.5.1 Experimental Setup . . . 120
5.5.2 Simulation Results . . . 121
5.5.3 Hardware Results . . . 122
5.6 Unoccluding Objects Via Manipulation . . . 124
5.6.1 Experimental Setup . . . 125
5.6.2 Hardware Demonstration . . . 128
5.7 Summary . . . 129

6 Discussion 135
6.1 Importance of Good Trajectory Initialisation . . . 135
6.2 Analysing the Behaviour of HAP . . . 137
6.3 Using Easy-To-Find Information to Bootstrap Hard-To-Find Information . 141
6.4 Manipulation for Perception . . . 142

7 Conclusions and Future Work 145
7.1 Thesis Summary . . . 146
7.1.1 Hausdorff Approximation Planner (HAP) (Chapter 3) . . . 146
7.1.2 Targeted Information Gathering (Chapter 4) . . . 147

7.1.3 Decentralised Coordination Framework (Chapter 5) . . . 147
7.1.4 Discussion (Chapter 6) . . . 148
7.2 Summary of Contributions . . . 148
7.2.1 HAP Framework . . . 148
7.2.2 Targeted Information Gathering VI Metric . . . 149
7.2.3 Active Perception Framework . . . 149
7.2.4 Decentralised Coordination Framework . . . 150
7.2.5 Theoretical Analysis . . . 150
7.2.6 Empirical Results . . . 150
7.3 Future Work . . . 151
7.3.1 HAP Framework . . . 151
7.3.2 Targeted Information Gathering Framework . . . 153
7.3.3 Decentralised Coordination Framework . . . 154
7.4 Outlook . . . 155

Bibliography **159**

Appendices **184**
A Experiment Supporting Figures . . . 185
A.1 Naive Task Region Base Allocations for HAP Experiments . . . 185
A.2 Apple Trellis Configurations for Harvesting Experiments . . . 186
B Supplementary Algorithmic and Implementation Details . . . 188
B.1 Null-Space Control for Avoiding Joint Limits . . . 188
B.2 Upper Confidence Bound 1 Applied to Trees (UCT) . . . 189
B.3 Discounted UCT (D-UCT) . . . 189

List of Figures

1.1 Examples of robots operating in semi-structured environments. 4
1.2 Simple illustration of Hausdorff Approximation concept. 8
1.3 Illustration of Hausdorff Approximation mapping from task space to configuration space. 9
1.4 Two robot arms coordinate their actions to discover and reconstruct apples on a trellis. 16

3.1 Example bookshelf environment illustrating graph construction. 42
3.2 A side-by-side comparison of HAs with largest coverage. 55
3.3 Visualisation of HA mappings for UR5. 56
3.4 Visualisation of HA mappings for UR5 with mobile base. 57
3.5 Bookshelf object configurations used for the simulation experiments. 63
3.6 Planning success rate and times for 7-DOF Sawyer experiments with the fixed-base model. 67
3.7 Planning success rate and times for 7-DOF UR5 with the mobile-base model. 68
3.8 Task-space control experiment setup for HAP evaluation. 71
3.9 HAs computed for UR5 with static base for task-space control experiments. 73
3.10 Subspaces computed for HAP-Mobile. 74
3.11 UR5 task-space control experiment results. 75
3.12 Our autonomous pruning system with a custom cutter mechanism. 76
3.13 Our pruning robot setup, consisting of a UR5e arm, an Intel RealSense D435 RGB-D camera, and custom pneumatically-actuated shears. 77
3.14 The test setup used for experimental evaluation. 77
3.15 Simulation visualisation of cutpoint generation. 79
3.16 Robot pruning a branch with simulation environment and in-hand camera view. 81

4.1 Searching for objects of interest, here apples, that are sparsely located and in some cases heavily occluded. 84
4.2 Ray tracing visualisation. 87
4.3 Visualisation of view-point evaluation function. 90
4.4 Two separate octrees (shown here as quadtrees for simplicity) used for the exploration and targeted information VIs. 94
4.5 Experimental setup for apple harvesting experiments. 97
4.6 Apple harvesting execution pipeline and subsystem layout. 98

4.7 Discretised viewpoints computed by HAP that are used by the active perception framework to search over. . . . 101
4.8 An example sequence of views computed by the active perception component in the apple harvesting experiment. . . . 105
4.9 Resulting reconstruction and segmented apples. . . . 106
4.10 An example grasp sequence from the apple harvesting experiment. . . . 108

5.1 High-level overview of decentralised planning framework. . . . 114
5.2 Experimental setup for multi-robot active perception. . . . 118
5.3 Multi-robot active perception roadmap construction. . . . 119
5.4 Apple trellis configurations for simulated and hardware experiments. . . . 120
5.5 Samples viewpoints from dataset used for simluated experiments. . . . 122
5.6 Apple detection rates. . . . 123
5.7 Simulated experimental results. . . . 124
5.8 Hardware experimental results. . . . 125
5.9 Examples of obstructions in a real apple orchard. . . . 126
5.10 Experimental setup for unoccluding objects via manipulation. . . . 127
5.11 Poke obstruction example. . . . 128
5.12 Push obstruction example. . . . 129
5.13 Hardware demonstration of manipulator unoccluding objects for active perception. . . . 132
5.14 Final reconstructed point cloud with segmented apples. . . . 133

6.1 Side view of bookshelf environment showing comparison of naive planner's initial trajectory seed and HAP's. . . . 136
6.2 Example subset of a Naive-Full trajectory sequence for Sawyer bookshelf experiment. . . . 137
6.3 Example HAP-Full trajectory sequence for fixed-base model bookshelf experiment. . . . 138
6.4 Example subset of a Naive-Mobile trajectory sequence for mobile-base model bookshelf experiment. . . . 139
6.5 Example HAP-Mobile trajectory sequence for mobile-base model bookshelf experiment. . . . 140
6.6 Detected ROIs overlaid on apples trellis. . . . 141

7.1 Unseen bookshelf environment. . . . 152
A.1 Subspaces and base positions (manually defined) for Naive-Mobile in single- and batch-query experiments. . . . 185
A.2 Subspaces and base positions (manually defined) for the Naive-Mobile method in task-space control experiments. Design intention as in previous figure. . . . 186
A.3 Trellis configurations used for apple harvesting experiments. . . . 187

List of Tables

3.1 Variants used for the Sawyer experiments. 64
3.2 Variants used for the UR5 experiments. 64
3.3 Parameters used for experiments. 66
3.4 HAP validation results. 70
3.5 Results from pruning experiments. 80

4.1 Experimental results for apple grasping. 109
4.2 Classification of hardware experiment failures. 109

List of Algorithms

3.1 Online Planner 43
3.2 Generate Trajectory Library 49
3.3 Modified Dijkstra 50
3.4 Get Candidate 50
3.5 Update 51
4.1 MCTS rollout with targeted information gathering VI 95
5.1 Dec-MCTS rollout with targeted information gathering VI 115
5.2 Decentralised Coordination Framework 116
5.3 Dec-MCTS Overview 117

List of Theorems and Definitions

3.1 Problem (Task and motion planning problem) 41
3.2 Problem (Motion planning problem) 42
3.3 Problem (Task sequencing problem) 42
3.1 Definition (ϵ-Gromov-Hausdorff approximation (ϵ-GHA)) 47
3.1 Lemma (Runtime of *Get Candidate* (Algorithm 3.4)) 58
3.1 Theorem (Runtime of *Modified Dijkstra* (Algorithm 3.3)) 59
3.1 Remark (Runtime dependence on number of mappings) 59
3.2 Remark (Runtime with mobile base) 59
3.2 Theorem (θ is an ϵ-GHA) 60
3.2 Definition (Geodesics) 61
3.3 Theorem (ϵ-GHAs preserve geodesics) 61

4.1 Problem (Viewpoint plannning) 87

5.1 Problem (Decentralised viewpoint planning) 113
5.1 Theorem (Complexity of decentralised viewpoint planning problem) 117

Nomenclature

Acronyms & Abbreviations

AGV	autonomous ground vehicle
AUV	autonomous underwater vehicles
BIT*	Batch Informed Trees
Dec-MCTS	decentralised Monte Carlo tree search
DOF	degrees of freedom
D-UCT	discounted-UCT
ϵ-GHA	ϵ-Gromov-Hausdorff approximation
FOV	field of view
GPIS	Gaussian process implicit surface
HA	Hausdorff approximation
HAP	Hausdorff approximation planner
IG	information gain
IK	inverse kinematics
MCTS	Monte Carlo tree search
NBV	next best view
PBVS	position-based visual servoing
PRM	probabilistic roadmap
ROI	region of interest
ROS	robot operating system
RRT	rapidly-exploring random tree

TAMP	task and motion planning
TSP	travelling salesman problem
UCB	upper confidence bound
UCT	UCB applied to trees
UR	Universal Robots
UTS	University of Technology, Sydney
VI	volumetric information

List of Publications

Conference Papers

1. **F. Sukkar**, G. Best, C. Yoo and R. Fitch, "Multi-Robot Region-of-Interest Reconstruction with Dec-MCTS," *Proc. IEEE Int. Conf. on Robotics and Automation*, May, 2019. **Winner of the IEEE Best Paper Award in Service Robotics.**

2. A, You, **F. Sukkar**, R. Fitch, M. Karkee and J. R. Davidson "An Efficient Planning and Control Framework for Pruning Fruit Trees," *Proc. IEEE Int. Conf. on Robotics and Automation*, May, 2020.

Journal Papers

1. **F. Sukkar**, J. Wakulicz, K. M. B. Lee and R. Fitch, "Motion planning in task space with Gromov-Hausdorff approximations", *The International Journal of Robotics Research*. ***submitted to**

Awards

Conferences

1. IEEE ICRA 2019 Best Paper Award in Service Robotics (Nominated for Service Robotics and Robotic Vision Awards)

University

1. UTS Faculty of Engineering and Information Technology 2019 Research Showcase - First Prize (Oral Presentation)
2. School of Mechanical and Mechatronic Engineering 2019 Research Showcase - First Prize (Oral Presentation)

Chapter 1

Introduction

Planning for robotic manipulators remains a challenge to this day despite its active research over the past several decades [1–10]. The word "planning" is general in a sense that it can apply to several aspects of a real-world system. For example, a robot searching for objects in an environment has to plan where it should place its sensor(s) to perceive the scene and then plan the sequence of motions to achieve this placement. Once objects are found the robot may wish to grasp them, hence the order in which the objects are to be grasped should be planned for and additionally the motions to achieve their retrieval.

Robot systems that carry out these tasks in the real-world have several algorithmic components and in order for them to perform effectively they must be tightly integrated and not treated separately. These components can include, yet are not limited to, active perception, object segmentation, task and motion planning, task-space control, and grasping.

This is particularly true of complex robotic platforms such as high-dimensional robotic manipulators with eye-in-hand sensors. Such systems are suitable for solving general problems that humans are capable of in challenging environments; for example, picking up objects off of shelves and tables [5, 11, 12], horticultural tasks such as fruit harvesting [13–16] and pruning [17], and searching for objects in clutter [18].

These general systems come at the cost of increased complexity in the context of both planning and perception. High-dimensional robot arms have highly non-linear motion models and the dimension of their state-space increases with the number of degrees of

freedom (DOF). Eye-in-hand sensors such as RGB-D (combination of a RGB image and its corresponding depth image) cameras have complex sensor models that are computationally expensive to compute. Current approaches make coarse approximations to these models in order to make the problem tractable [8, 14, 19–22]. This can lead poor and undesirable performance due to an inaccurate description of the system behaviour.

A key insight is that many approaches that resort to these approximations aim to be as general as possible and make little to no assumptions about the problem or environment. We argue that for many useful applications this is overly conservative, rather many problems tend to be semi-structured; that is, there is some underlying structure known beforehand in either the environment or tasks being performed.

Furthermore, in order to expand the applicability of these system we want to be able to scale them up to multiple robots in order to speed up throughput and potentially even carry out tasks that are not possible with only a single robot. A real world example demonstrating this potential of utilising multiple robots is the deployment of thousands of robots in Amazon warehouses [23]. Coordinating multiple robots effectively presents a computational challenge alone and achieving this with the complex systems described above is even more challenging.

This thesis is concerned with solving such problems in a robust and efficient manner whilst retaining real-time performance which is essential for deploying these systems in the real world and performing valuable work. In order to achieve this we need to be able to faithfully describe the behaviour of system. In particular, we want to avoid making coarse approximations to various components of the system, such as the motion and sensor models.

The robot systems described above tend to be faced with the problem of searching for objects in complex environments. As such, another challenge this thesis addresses is to effectively balance between exploration of the scene and targeted observation of regions we're interested in. Lastly, if we want to use these systems in the real world, potentially around other humans, they must exhibit predictable behaviour. Key to achieving this is by exploiting prior structure in the problem, a characteristic of semi-structured environments, to attain practical computational guarantees.

1.1 Planning In Semi-Structured Environments

There is a spectrum of environments in which robots operate in, for example robots operating in industrial settings even to this day are fully pre-programmed and are able to carry out all the tasks they need without ever having to change their program [2, 4, 24]. This is possible when the environment is known completely beforehand and nothing is expected to change. In these settings robots, for example industrial manipulators, are placed inside a physical cage and no humans are allowed to enter while the arm is operating.

On the other hand, applications exist where the environment is completely unknown and no assumptions can be made prior to operation. For example, off-planet rovers [25] and autonomous underwater vehicles (AUVs) [26] have to deal with these environments. In these environments general methods are required in order to handle all potential scenarios. This needs to be considered by all subsystems of the robot, including motion planning, perception algorithms, sensing, localisation, and control.

However, there exists many applications that lie somewhere in between these two extremes. We refer to these problems as semi-structured since there exists some prior knowledge about what to expect in the environment or task beforehand. For example, a collaborative robot (cobot) may know the location of static obstacles in the environment and the volume in which it needs to carry out some tasks; however, it may need to react quickly and safely to a human operating around it [27]. Another example is a manipulator grasping objects from shelves and cabinets in a domestic or warehouse environment [5, 6, 12, 28–30]. The shelves are static and their dimensions can be measured beforehand; however, the objects on the shelves might not be known and their locations can change. Horticultural applications such as robotic pruning [17, 31] and fruit harvesting [13, 15, 21, 32] are another example of semi-structured problems, as shown in Figure 1.1. In these settings the crops are usually organised into structured rows; however, the location of their fruit or cut points are not known a priori.

As such, planning for emerging industrial applications of robotic arms must support a greater degree of autonomy than has traditionally been necessary. These applications share a need for agility and rapid deployment that differs substantially from traditional

(a)

(b)

Figure 1.1: Examples of robots operating in semi-structured environments.

applications of industrial manipulators, which typically are characterised by repetitive motions in highly structured environments with planning performed offline. General planning algorithms that make no assumptions about the environment are often limited in their ability to reliably produce feasible trajectories online in high-tempo industrial tasks. We are interested in developing new planning methods in settings where prior knowledge can be exploited to satisfy practical performance goals without loss of common existing analytical guarantees.

1.2 Manipulator Planning for Perception

Perceiving the world using on-board sensors is crucial in order for robots to carry out meaningful tasks in the real world. Examples of this include using sensor information to detect the presence [33, 34] and pose of objects [35, 36] for manipulation [15, 37–39], perform inspections [40], build a map to navigate through [41], or reconstruct an environment to avoid colliding into obstacles [42].

The quality of the sensor data may be highly dependant on the viewpoint it was captured from. For example, from one viewpoint an object may be occluded, whereas from another it may not. Therefore, blindly moving the sensor around and hoping for good data in these scenarios is not an effective approach. This motivates the problem of *active perception* [43, 44] where the goal is to plan a sequence of viewpoints with some objective in mind, such as gathering information, while satisfying motion constraints of the agent.

In many applications this involves a mobile robot, such as an autonomous ground vehicle (AGV), with a fixed sensor or a robotic manipulator with an eye-in-hand sensor moving around the scene in order to perceive its environment and carry some downstream task. This is not unlike what humans do in every day life to detect and manipulate objects. In a sense we are trying to equip robots, in particular manipulators, with the ability to perform such tasks in much the same way that humans do. This is a shift away from traditional applications of robot arms which are typically found in the real world in industrial settings where the environment is completely known beforehand and the robot carries out the same tasks repeatedly [4].

While active perception is a well studied problem [44], its application to robotic manipulators is less common in comparison to other robotic platforms. There has been growing interest in expanding the use of robot manipulators into applications where the environment and tasks can change dynamically [45]. We argue that active perception is necessary for robotic manipulators to be effective and safely operate in these applications.

Importantly, humans are able to carry out such tasks in a fast, reliable and efficient manner. Fast in the context of a robot means that it should be able to compute an executable plan in real-time. Typically humans can reason about simple tasks and react within a few

hundred milliseconds [46] and as such an aim of this work is to achieve similar planning times. Reliable means that these plans are computed with a high success rate and produce intuitive and predictable behaviour. Efficient means that the plan to be executed results in smooth trajectories that avoid unnecessary movement of the arm. This thesis is concerned with achieving these desired characteristics, particularly in the context of manipulator perception tasks.

1.3 Limitations of Current Manipulator Planners

High-dimensional manipulators are an effective choice for the problems described above because they can look around occlusions and even potentially reach around obstacles to grasp objects. The robot arms we are interested in are at least six-DOF since this is necessary to achieve an arbitrary end effector position in 3D space with a specified orientation [47].

Fast, reliable and efficient have traditionally been conflicting objectives for manipulator motion planning. For example, feasible sampling-based planners reduce computation time at the cost of path optimality [48–51]. Trajectory optimisers produce fast and efficient trajectories yet are susceptible to local minima [52–54]. Grid-based planners produce efficient trajectories reliably with appropriately chosen parameters yet their complexity increases exponentially in the search space's dimension [55, 56].

The difficulty of manipulator planning is further compounded when multiple tasks need to be planned for and sequenced. For high-DOF manipulators, many different configurations can be utilised to perform the same task. Hence to determine an optimal sequence, all possible trajectory sequences and task configurations need to be considered. Given that the problem of computing a single feasible trajectory is computationally expensive alone, to compute multiple trajectories and determine an efficient sequence in feasible time there is a need for an alternative method.

Naive methods use approximate sequencing metrics such as Euclidean distance between tasks in the workspace of the robot [14, 21]; however, this may heavily underestimate the true cost of trajectories, particularly in complex environments [57]. Additionally,

naive sequencing methods can result in manipulators approaching joint limits or awkward configurations causing highly sub-optimal planning for subsequent tasks. This could be considered in the task sequencing objective; however, there is no obvious choice in how one should formulate such an objective while maintaining efficient trajectories.

A common objective of most planners is to be as general as possible and require little to no assumption about the environment and tasks being carried out by the robot. However, there is an abundance of robotic problems where this is not necessary and hence such an assumption poses unnecessary challenges on the planning algorithm. In semi-structured environments we tend to have at least some knowledge of what to expect in the environment.

If we can construct these special cases of what to expect, how can we leverage them to make planning fast, reliable and efficient without being overly restrictive on how much the environment can change? This thesis is concerned with proposing a solution that effectively trades off between a general planner with no assumptions and one that has complete knowledge about the task and environment. Given a planning method with these characteristics, we can facilitate higher level planning for tasks, such as multi-robot active perception, that would otherwise not be possible using general planners.

The idea of leveraging domain knowledge is not new and recently there has been a lot of success in learning based methods. For example, recently learning based methods have replaced parts of traditional sampling-based planners with a neural network that was trained on representative datasets [58, 59]. In a similar vein, learning based methods have been proposed for choosing the next best view in active perception applications [60, 61]. However, these methods have their drawbacks, for example it is difficult to infer their behaviour across the entire distribution of problem instances and they can be timely to train.

Instead we propose a planning method that leverages domain information and has analytical guarantees about its behaviour. This is important in applications where predictable behaviour is crucial, for example when cobots are working closely with humans. Furthermore, instead of making coarse approximations to the robot's motion model our method

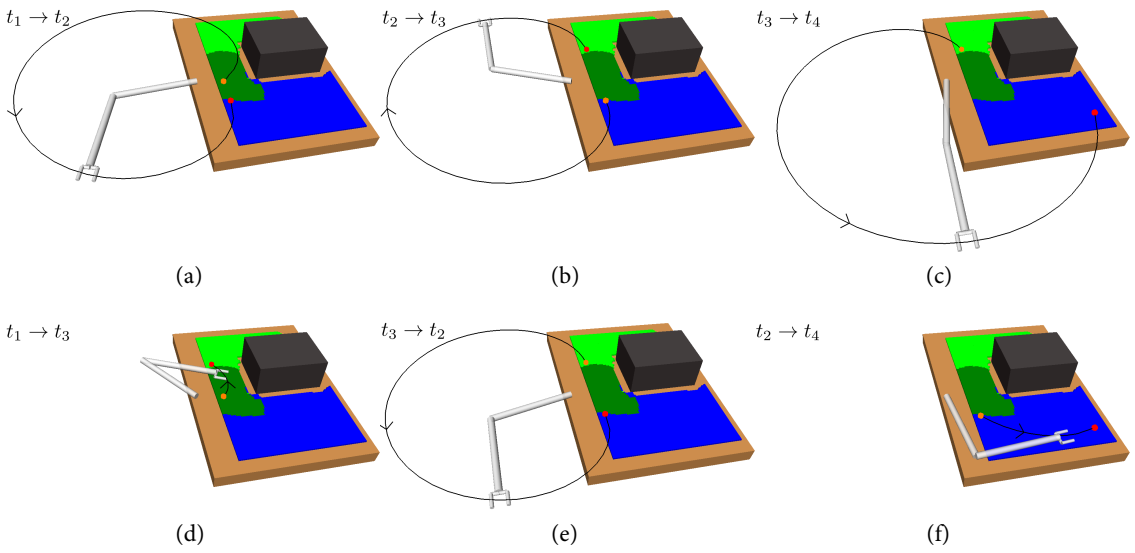


Figure 1.2: A two-DOF robotic manipulator tasked with moving its end effector to four unordered positions ($t_1 - t_4$) on a table-top environment with a box obstacle (dark grey). Positions are shown as coloured dots. The end-effector trajectory is drawn with direction of movement indicated by the arrowhead. Green and blue regions are subspaces within which end-effector motion does not require large changes in configuration. Overlap is shown in dark green. (a)-(c) show a sequence of trajectories produced by a naive planner. (d)-(f) show the sequence produced by our HAP method. HAP's choice of sequencing exploits short within-subspace trajectories in (d) and (f), whereas the naive planner's choice of initial position and configuration results in long trajectories to satisfy kinematic constraints.

enables division of task and motion planning into two separate problems whilst still faithfully describing the behaviour of the system. This greatly simplifies the problem, enabling very fast and reliable planning over long task sequences.

1.4 Planning with Hausdorff Approximations (HAs)

To illustrate the issue when using a coarse approximation for the motion of the arm for planning over multiple tasks, consider the scenario in Figure 1.2. Here, a two-DOF robot arm is tasked to move to three positions on a table with a grey box obstacle. The top row is the sequence produced by a naive approach which uses workspace distance between these positions as a heuristic for task sequencing. As can be seen, even in this simple 2D scenario using a coarse heuristic can result in large wasted motion.

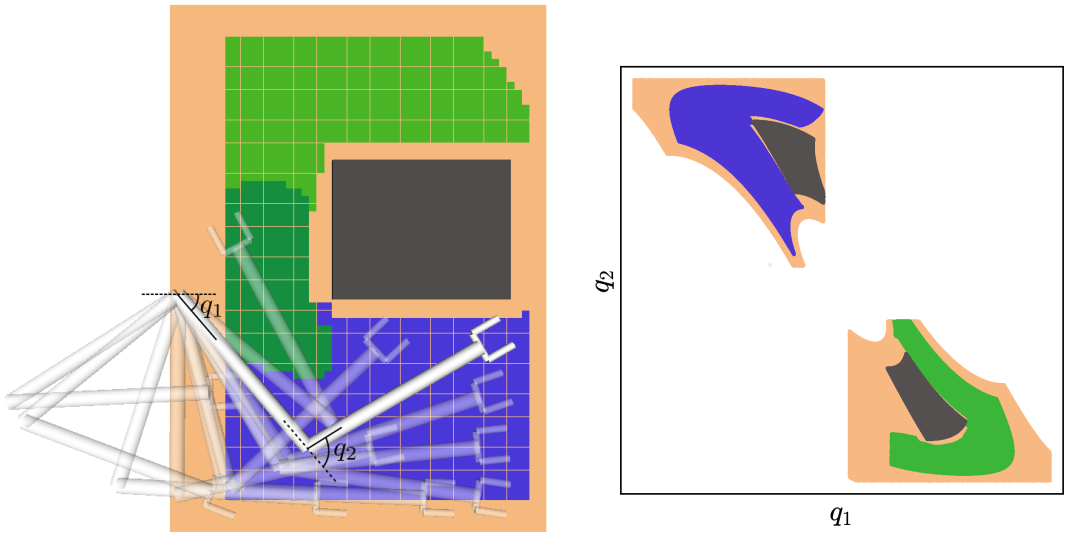


Figure 1.3: Illustration of two different Hausdorff Approximations (green and blue) induced by an obstacle (dark grey) for a planar two-DOF arm operating in a table-top (light brown) environment. (a) shows the subspaces in task space (note the overlapping region in darker green which are positions reachable from both "elbow-up" and "elbow-down" configurations) and (b) shows their corresponding mapping in configuration space. In configuration space the subspaces are clearly separated and all the positions within the same subspace are easily reachable from each other, requiring no large joint movements.

Differently, the bottom row is the sequence produced by the planning method proposed in this thesis called Hausdorff approximation planner (HAP). Clearly, the sequence produced by HAP is more efficient since there is significantly less arm motion. HAP uses what are called ϵ-Gromov-Hausdorff approximations (ϵ-GHA), or Hausdorff approximations (HA) for short, to achieve fast, reliable and efficient planning over long task horizons. HAs are a mapping between metric spaces, task and configuration space in this context, that ensure distances between points in one space are approximately preserved in the other.

In Figure 1.2 the workspace is overlaid with two distinct regions, shown as green and blue, and an overlap of these two regions, shown as dark green. Within these regions the arm can move around comfortably without having to perform a large change in its joint angles. These regions are the HAs and approximately preserve distances when mapping between the task space, positions in workspace for this example, and configuration, or joint, space. In the naive sequence this information is lost in the workspace distance heuristic and as a consequence the arm performs unnecessary switching between these two regions. As such,

HAs are particularly useful for high dimensional robots where it is possible to perform tasks in more than one way.

An alternate view of these HAs is seen in Figure 1.3b. As can be seen, in configuration space the HAs map to two distinct and separate subspaces. To move between these subspaces a large path has to be taken in configuration space even though the path distance may be small in task space. For higher-DOF arms, the set of manipulator trajectories moving through the task space in a "shoulder-in" configuration versus a "shoulder-out" configuration would be another example of two differing subspaces.

The HAP framework will be described in detail in Chapter 3, including how these HAs are computed for general robot manipulator kinematic configurations, even the mobile base case, and then how to leverage these for effective online planning. Importantly, in the online scenario HAP can deal with changes in the environment and tasks can be different to those used to construct the HAs.

HAP utilises what is known as a trajectory library representation [62–64] and generates a set of trajectories that span the HAs. These trajectories are stored in a library that can be retrieved and modified online to adapt to changes in the environment and task space definition.

Furthermore, the HAs can be tailored to minimise any given objective function. This can be useful for incorporating non-trivial application specific costs such as those related to the task being carried out by the manipulator. Additionally, the task sequencing problem is significantly reduced since the HAs resolve kinematic redundancy and allow for reliable concatenation of trajectories over long task horizons.

A theoretical analysis will be provided which proves that the mappings found by HAP are indeed HAs and as a consequence shortest paths in task space are approximately shortest paths in configuration space. In other words, trajectories through these HAs are guaranteed to be of bounded length and this bound can be set via an intuitive tuning parameter. This is important in the applications we are interested in where predictable and intuitive behaviour is highly desirable, particularly when operating around humans.

1.5 Scope

As described above, the problems we are interested in solving in this thesis cover several research areas. The scope of research areas covered and the intended problem settings will be described in this section.

1.5.1 Manipulator Task and Motion Planning

Manipulator motion planning refers to computing a sequence of motions in configuration, or joint, space to move it from a start configuration to a goal configuration [65, 66]. We consider additional constraints such as avoiding obstacles and physical constraints of the system such as singularities and joint limits. Task and motion planning refers to planning a sequence of trajectories to satisfy some tasks while considering the low-level motions of the robot [67].

Task and motion planning in the literature often considers additional constraints, such as task feasibility [68]; however, in this thesis the only task level constraints we consider are achieving a given goal pose, remaining within joint limits and avoiding collisions. However, for some tasks, for example task-space control, constraints such as avoiding joint limits and singularities can be implicitly considered via a cost function in the HAP framework. As such, other more suitable terms for this problem setting exist in the literature such as task sequencing [69–72] and multi-goal path planning [73, 74]. In the context of planning with HAs we refer to this problem as motion planning in the task space.

Task sequencing for manipulators is difficult because the complexity of computing an optimal sequence is exponential in the number of tasks and grows with the number of inverse kinematics (IK) solutions for each task. The number of possible sequences is order $O(|Q(t)|^{|T|})$ where $|Q(t)|$ is the cardinality of the largest set of IK solutions for a single task $t \in T$, and $|T|$ is the number of tasks.

In this thesis we wish to find an alternative method that significantly reduces this complexity without resorting to coarse approximations, such as Euclidean workspace distance, which can heavily underestimate the true cost of the trajectory sequence. Furthermore,

we wish to ensure the trajectories themselves are efficient and intuitive with some level of predictability.

1.5.2 Information Gathering and Active Perception

As touched on earlier, perceiving the environment passively without any feedback may not be suitable for certain scenarios. For example, in highly cluttered environments the quality of the sensor data is potentially highly dependant on the viewpoint, for example due to occlusions. In these problem settings the robot cannot exhaustively observe the environment. Rather, a resource budget, such as time or energy, is given and the robot must do the best it can given this constraint.

"The best it can do" is usually quantified via an objective function. The applications this thesis is concerned with are information gathering problems, such as reconstructing objects of interest in a scene. For these problem settings a reward function, or information measure, defines the value of an action, such as taking a sensor scan from a particular viewpoint, with respect to the objective of the perception task [75–77]. The intent is that this reconstruction can then be utilised by some downstream task, such as object segmentation and/ or manipulation.

In this thesis we aim to formulate a framework that solves such problems effectively and is flexible enough to be applied to a wide range of difficult problems and robotic platforms, such as high-dimensional manipulators with complex sensor models. We focus on developing a reward function that enables effective performance of these systems and considers the accurate cost of these real physical systems while retaining real-time computation.

1.5.3 Task-Space Control and Manipulation

A general-purpose robotic system should not only be able to perceive the environment, it should be able to manipulate and interact with it too. In order for a robotic manipulator to do so it should be able to account for uncertainties in sensor measurements. These measurements may be from scans of the environment or encoder information from the joints of the robot arm to estimate the location of the sensor with respect to its base.

Task-space control methods [78], such as visual servoing, aim to solve such problems by incorporating feedback into their control loop; for example, as a robot arm moves its end-effector towards an object in order to grasp it. This feedback is usually in the form of sensor measurements which are then used to update the estimated pose of the target.

However, due to their locally optimal nature these methods can fail due to physical constraints such as approaching joint limits and singularities [79, 80]. We show that task-space control is tightly integrated with motion planning and these failures can be avoided if they are treated together as a dependant system. As such, this thesis is concerned with first integrating task-space control and manipulation into a general framework and additionally improving the performance of these control schemes by avoiding failures.

1.5.4 Multi-Robot Active Perception

Scaling these approaches to multiple robots unlocks the potential for higher throughput and carry out tasks that would not otherwise be possible with a single robot. As such, these systems are in a sense the most general and have the potential to be as effective as humans are in carrying out tasks.

While these systems are desirable, from an algorithmic perspective they present several challenges. One of the main challenges is that the search space grows exponentially in the number of robots. Most of the work in the multi-robot planning literature aims at overcoming this computational complexity while retaining effective performance. Deploying these systems in the real world introduces additional challenges such dealing with limited communication infrastructure and risk of one or many robots in the team failing.

Multi-robot planning algorithms can be either centralised or decentralised [81]. The latter is desirable in real world settings since it can better deal with the issues described above [82, 83]. Since each robot computes its own plan, if one robot fails it is not catastrophic and the rest of the system can continue carrying out their objective regardless, adding a layer of redundancy. Furthermore, robots incorporate the other robots' plans through communication asynchronously when available. Hence, these systems can deal with intermittent communication and their performance degrades gracefully.

However, multi-robot systems are notoriously difficult due to the fact that bad coordination can lead to worse performance than if an uncoordinated system were used to carry out the same task [84]. As such, great care and importance must be placed on the effectiveness and robustness of the coordination algorithm, particularly with respect to unreliable communications.

In this thesis we aim to develop a decentralised coordination framework general enough to carry out arbitrary perception tasks while exploiting any known underlying structure of the problem. Furthermore, the framework should be able to handle the complex systems we are interested in and achieve effective performance that scales with the number of robots while retaining real-time operation and being robust to intermittent communication.

1.5.5 Problem Settings

1.5.5.1 Robust manipulator planning over long task horizons

Many real-world practical problems involve carrying out many tasks one after the other. For example, picking up objects off of a bookshelf involves placing the robot's end effector in and around the shelves repeatedly. Each of these motions is not independent of each other, particularly for a high-dimensional robot arm which can perform a task in many ways and has physical constraints, such a joint limits. The way in which the robot chooses to carry out these tasks highly depends on the ordering and assigned arm configurations for each task.

We are interested in solving such problems and demonstrate the efficacy of our method using several different simulated and real world scenarios with a diverse range of robotic manipulator platforms in Chapter 3. Furthermore, we focus on semi-structured problems settings and show how exploiting the underlying structure in the problem can lead to significantly better performance than naive methods.

1.5.5.2 Searching for objects in clutter

Searching for objects in clutter is a challenging yet important problem to solve since it can be applied to many useful real-world problems, such as robotic harvesting and search and rescue missions. These settings exhibit many of the challenges we are interested in, including looking around occlusions and balancing between exploration and targeted observations of regions of interest.

The problem setting involves sparsely located and partially occluded objects that will need to be reconstructed well enough for some downstream task, such as object segmentation or object grasping, given a resource budget. No prior model of the scene is known other than the bounding volume in which the objects of interest are within; however, we seek a method that can support and exploit a prior model for better performance if one is available.

To solve this problem, in Chapter 4 a non-myopic active perception framework will be developed that considers the complex motion and sensor model of a high-dimensional manipulator with an RGB-D eye-in-hand sensor. We develop a novel reward function that is used for viewpoint planning and leverage HAP for generating realistic trajectory cost estimates for effective budgeting.

In contrast to problem formulations such as map building and full scene reconstruction, we consider a formulation that requires targeted collection of information about unknown regions of interest (ROIs). Targeted information gathering of ROIs is guided by a viewpoint evaluation function that predicts both how many ROIs may be discovered and how much they improve our model of previously discovered ROIs, quantified as volumetric information (VI) [85]. A non-myopic planner computes a sequence of viewpoints that maximises the expected information gain (IG) over a long time horizon while satisfying budget constraints for the action costs.

1.5.5.3 Manipulating objects

Tasks such as robotic pruning and harvesting require perception in order to be effective. First of all, if the location or number of objects to be manipulated in the environment are

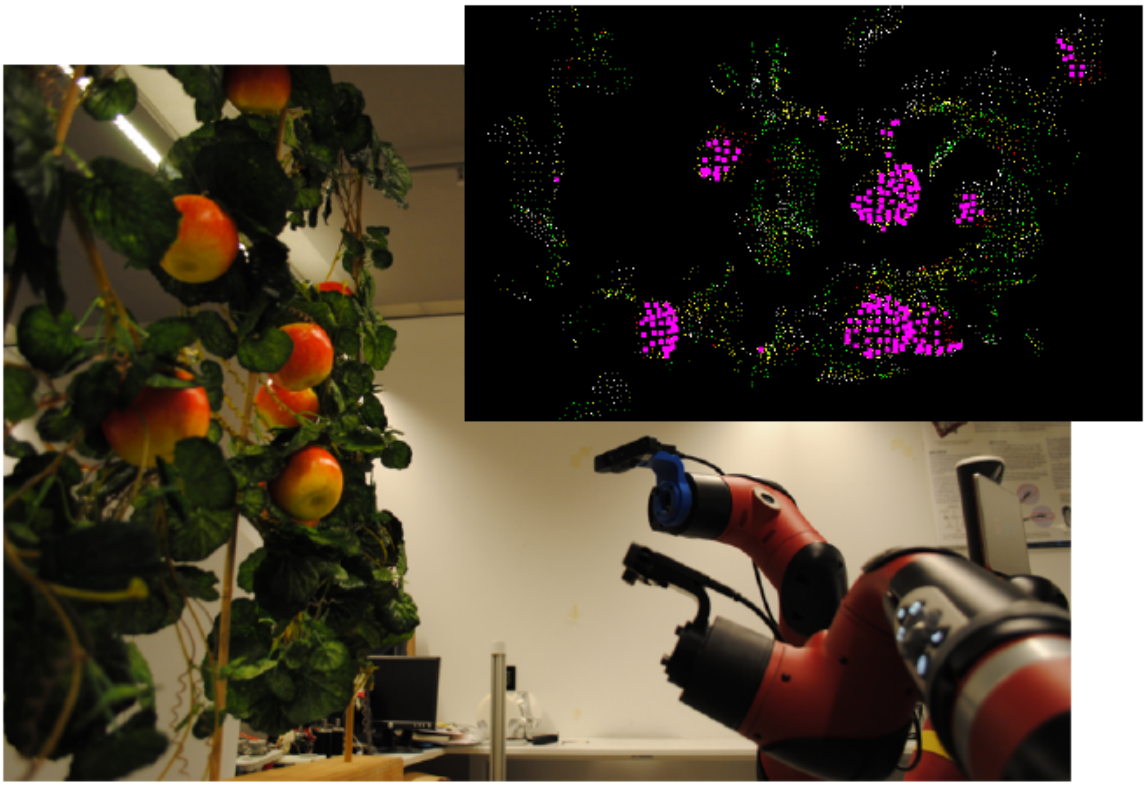


Figure 1.4: Two robot arms coordinate their actions to discover and reconstruct apples on a trellis. *(inset)* Resulting point-cloud and segmented apples.

not known then they need to be reconstructed and segmented from this reconstruction. Furthermore, the initial estimates of their location may be noisy and uncertain. Therefore there needs to be closed loop feedback from a sensor that corrects for this. Additionally, the control of the arm needs to account for this feedback and correct its trajectory accordingly.

This type of task-space control is known as visual servoing. Basic visual servoing is trivial; however, several failures can occur during the approach of the arm. For example, the arm may reach joint limits or hit a singularity during its approach. We demonstrate the flexibility of HAP, developed in Chapter 3, to optimise the HAs to favour arm configurations that are far away from its joint limits and have high manipulability which results in fewer failures. This is incorporated into the active perception framework in Chapter 4 to form a general-purpose framework for robotic manipulation in complex environments.

1.5.5.4 Decentralised coordinated planning

Scaling the active perception framework to multiple-robots allows for greater throughput and more general systems that can solve a greater range of problem. In Chapter 5 we propose a decentralised coordination framework that deals with several of the real world problems that can arise when deploying multiple robots and scales well with increased team sizes. Importantly, this framework incorporates HAP, from Chapter 3, and the active perception framework, from Chapter 4, to enable the complex motion and sensor models we are interested in while retaining real-time performance. Differently, the viewpoint planner now plans sequences of viewpoints for a team of robots that maximises the joint expected IG while satisfying budget constraints for each individual robot.

We demonstrate this on a scenario where two robot arms with eye-in-hand RGB-D sensors collaboratively search for objects of interest in a cluttered environment, illustrated in Figure 1.4. Additionally, preliminary experiments are performed to demonstrate the flexibility of the framework where parts of the environment need to be manipulated in order to reveal objects of interest.

1.6 Principal Contributions

The main contribution of this thesis is a collection of component algorithms that enable general-purpose robot systems for solving real-world problems, with focus on manipulator perception tasks exhibiting complex motion and sensor models. The specific contributions are enumerated as follows:

- A novel robotic manipulator planning framework called **Hausdorff approximation planner (HAP)** that utilises prior knowledge for building a library of trajectories which can be concatenated reliably over long task horizons, adapted quickly online and queried for accurate costs in constant time. Automatically generating smooth,

short trajectories online directly from a user-defined task space with reasonable performance guarantees is a step beyond current general-purpose motion planning methods, which generally require expertise to design implementations that perform well in practice.

- A novel **targeted information gathering reward function** for effectively balancing between exploration of a scene and targeted observation of regions of interest when searching for objects in cluttered environments. Our formulation enables very fast evaluation which scales well with complex scenes, allowing for use within non-myopic planning frameworks and even multi-robot systems.

- A **real-time, non-myopic active perception framework** that integrates the above components to enable complex motion and sensor models, such as high-dimensional robot arms with eye-in-hand RGB-D sensors, and solve a wide range of perception tasks. Easily perceived interest indicators are used to guide sensor data collection to improve a more complex perception task, such as object recognition; this approach leads to a fundamental improvement in efficiency.

- A **decentralised coordination framework** that generalises the above active perception framework to multi-robot problems, such as collaboratively searching for objects in clutter. The framework is capable of solving general multiple robot problems, even those that require manipulation of the environment within the active perception pipeline.

- A **theoretical analysis** showing that the HAs found by HAP are indeed an ϵ-Gromov-Hausdorff approximation (ϵ-GHA) and trajectories through these are guaranteed to be of bounded length and approximately correspond to geodesics in the task space. This is an important result as it encourages predictable and intuitive behaviour. Additionally, a complexity analysis of the HAP algorithm and the decentralised viewpoint planning problem is given.

- **Empirical results** from demonstrations of each of the above on several different applications both in simulation and hardware, including robotic pruning, robotic harvesting, and searching for objects in clutter using multiple robotic manipulators with eye-in-hand sensors. These results validate the theoretical claims and evaluate

the behaviour of the proposed algorithms on a diverse range of robotic platforms and problem settings.

The work presented in this thesis has partially appeared in our previous publications. More specifically, Chapter 4 and 5 are based on [16, 18]. The HAP framework in Chapter 3 is a major extension of work appearing in my Masters' thesis [57]. Some of the motivating text in Chapter 1 and related work, on motion planning specifically, in Chapter 2 are also an extension of text appearing in [57]. The work in Chapter 3 and its discussion in Chapter 6 are currently being prepared for submission as a journal paper to IJRR under the title "Motion planning in task space with Hausdorff approximations". The pruning experiment in Section 3.8 is from [17]. Fouad Sukkar was a main contributing author in all of these publications.

1.7 Thesis Outline

The thesis is organised as follows:

Chapter 2 explores relevant literature and state of the art for motion planning, task planning, next-best-view planning, information gathering and multi-robot active perception. In particular, the gaps in the state of the art and their relation to the research contributions of this thesis are highlighted.

Chapter 3 presents HAP a task and motion planning framework designed for high-dimensional manipulators operating over long task horizons in semi-structured environments.

Chapter 4 details a targeted information gathering algorithm and integrates it with HAP to form a general active perception framework, suitable for eye-in-hand manipulator platforms.

Chapter 5 generalises the active perception framework in Chapter 4 to a decentralised coordination framework suited for multi-robot active perception applications.

Chapter 6 discusses the results from Chapters 3-5 and analyses algorithmic choices and implementation details.

Chapter 7 concludes and summarises the thesis and explores ideas for algorithm improvements and future work.

Appendix A provides supporting figures for the experiments in Chapters 3-5.

Appendix B provides supplementary algorithmic and implementation details for the experiments in Chapters 4-5.

Chapter 2

Related Work

The work in this thesis covers a wide range of research areas, including task and motion planning, coverage planning, next-best-view planning, information gathering, and multi-robot active perception. The following sections describe relevant research in each of these areas, their state of the art and gaps in the literature that this thesis aims to improve upon. To begin with, however, a discussion on planning methods that utilise domain knowledge is given which extends on motivating points given in the introductory section.

2.1 Planning with Domain Knowledge

The idea of leveraging domain knowledge for improving planning has been explored extensively in the literature. Learning based methods are one prominent area, for example, recently MPNet [58] and PlannerFlows [59] by Qureshi and Lai et al. were proposed which replace parts of a traditional sampling-based planner with a neural network that was trained on representative datasets. Their methods were compared with state-of-the-art sampling based planners and achieved lower computation times and path costs. In [86] Terasawa et al. propose to learn a heuristic map that generalises over similar environments using a 3D CNN and integrates this within the Task-Space RRT (TS-RRT) planning framework [87]. However, these methods have their drawbacks, for example it is difficult to infer

their behaviour across the entire distribution of problem instances and they can be timely to train.

Another approach that leverages experience are trajectory library methods which precompute a library of trajectories and adapt them in real-time to satisfy online tasks. For example in [62, 88] Lin et al. utilise a trajectory library along-side a discrete planner for planning complex humanoid motions in unstructured environments. Combing these two methods improved planning success, particularly in scenarios where the branching factor of the search space greatly increases, such as when traversability becomes difficult and utilising palm contacts becomes necessary.

A similar idea was explored in [64] and [63] by Lee and Ellekilde et al. for robotic manipulators where a library of trajectories is precomputed over a roadmap of representative end-effector poses. Their methods achieved fast online planning times; however, success rates and trajectory costs were not consistent. This is partly due to the online adaption process which did not consider collision avoidance. Moreover, these trajectory library methods tend to not consider trajectory cost optimisation in the library construction which can lead to arbitrary performance when adapted online.

In [18] and [17] Sukkar et al. used a planning framework called FREDS-MP [57] that builds on the ideas of [64] and [63]; however, it allows for obstacles to be different in the online scenario and factor them in when adapting the library trajectories. Additionally, the library trajectories were optimised to be close to each other in configuration space and structured in a way to facilitate reliable task sequencing. While promising results were achieved for a redundant seven-DOF manipulator, performance decreased substantially when applied to a non-redundant six-DOF. This was mainly due to the six-DOF requiring a trajectory library representation that supports multiple subspaces in order to cover its task space and retain closeness across trajectories, which was not supported by the framework.

The HAP framework proposed in this thesis is an extension to FREDS-MP which generalises to multiple subspaces. In doing so it allows for greater application flexibility; for example, non-redundant systems and even mobile robots are now suitable within the framework. Furthermore, we provide a bounded suboptimality theoretical analysis which

guarantees that the difference in length between geodesics in the task space and their corresponding library trajectory is bounded through the utilisation of HAs.

Riemannian Motion Policies (RMPs) [89] are a type of motion generator that solves the motion planning and control problem. However, from a high level they can be seen as a compact and efficient representations of cost functions that capture the geometry of the task space. The trajectory library generated by HAP can be interpreted as a less general yet more efficient representation that is tailored for sequencing multiple tasks and global planning rather than local reactive control. While RMPs require forward integration for trajectory generation, HAP can leverage the library trajectories for initial guesses and their costs are provided at constant time. In [89] Cheng et al. attempt to overcome the issues associated with local planning over long task horizons by formulating retraction heuristics. An example of this is when the robot is in an outstretched position and needs to reach another outstretched position; however, there is an obstacle between them such that the local controller fails. HAP implements a similar retracting behaviour when moving between subspaces; however, instead of having to deal with potentially complex heuristics HAP can directly inform when this retraction is necessary.

The trajectory library generated by HAP can provide diverse solutions to achieve the same task. A similar idea was explored by Osa et al. in [90] which proposed an algorithm called Stochastic Multimodal Trajectory Optimization (SMTO) for finding multiple trajectories corresponding to the modes of any given objective function. While their method was capable of finding diverse solutions it focuses on a single given task. In this work we aim to find diverse solutions that cover a given task space which is more useful when planning for multiple tasks. Furthermore, SMTO takes several seconds to compute solutions for a single task. In this work we are aiming to produce solutions to several tasks within a few hundred milliseconds each.

Finding subspaces in the configuration space and exploiting them for planning was explored by Reid et al. in [91]. They proposed a method called Hierarchical Bidirectional Fast Marching Trees (HBFMT*) which decomposes the configuration space into multiple subspaces and leverages these to bias sampling when computing a path solution. Their

method is aimed at high dimensional reconfigurable robots and achieved natural transitions between configurations when planning through complex environments. While our method utilises a similar concept, our goal is to find a set of subspaces that maintain bounded lengths in configurations space and cover a given task space. Furthermore, while their method decomposes the configuration space based on the kinematic structure of the robot our method can consider any given objective when computing the subspaces. For example, prior knowledge such as obstacle configurations and task specific costs can be incorporated. Additionally, our method provides a library of precomputed trajectories which can be adapted quickly online and are suited for planning over long task sequences.

2.2 General Motion Planners

While the related work in the previous section is more closely aligned with our work's direction we provide a review of more general motion planning algorithms here. Motion planning algorithms can be categorised into sampling based, grid-based search and trajectory optimsation. Each of these have sub-categories such as any-time, any-angle, incremental or a hybrid combination. Fundamentals on motion planning can be found in books by Lavalle [66] and Choset [65] and rich surveys of motion planning algorithms by Elbanhawi and Gammell et al. in [92, 93]. Here we will explore each of these categories and recent state of the art methods and discuss their strengths in addition to their shortcomings, particularly in the context of manipulator planning.

2.2.1 Grid-Based Planning

Grid-based search planners overlay a grid on to the robot configuration space, where each point in the grid corresponds to a robot configuration. The task is then to connect configurations in the grid, ensuring connections are in free configuration space, such that a path between the start and goal configuration is formed. One of many graph search algorithms can then be used to efficiently search through the grid for a solution. The grid can be constructed in various ways; however, its main tuning parameter is the grid resolution. A higher grid resolution will allow planning through narrower obstacle clearances; however,

this comes at the cost of increased search time. Furthermore, the number of nodes in a grid will increase exponentially in the configuration space dimension, making them non-ideal as a standalone planning method for high-dimensional configuration spaces.

Dijkstra's algorithm [55] is an early solution to finding the shortest path through a graph. Its minimum-priority queue implementation [94] is the fastest known single source, shortest path algorithm for arbitrary directed graphs with unbounded non-negative weights. A* [56] is more efficient when finding a path from a single source and goal. It is an informed search, or best-first search, algorithm which means that it will find the lowest cost path in the graph from start to goal configuration if an admissible heuristic is given.

D* and D* lite [95, 96] are incremental planning algorithms that were introduced to deal with dynamic graphs, i.e. those which change over time. Instead of simply re-planning from scratch, D* optimises the number of edge cost changes needed to maintain a lowest cost path. It was observed, however, that if the goal node changes then A* is more efficient.

Grid-based search algorithms traditionally suffered from the fact that they could only produce paths with heading changes that were constrained to the direction between subsequent grid points. This often results in sub-optimal, jagged paths. Hence, any-angle path planners [97–101] were introduced to allow turns in the path to have any angle when searching over a discrete graph, producing optimal or close to optimal paths. They are advantageous over traditional geometric planning methods, such as searching over a visibility graph, since the number of edges in the graph grows linearly with number of vertices, as opposed to growing by the number of vertices squared, resulting in faster computation.

2.2.2 Sampling-Based Planning

For high dimensional systems, such as manipulators, grid-based search motion planners are typically not used due to their computational complexity which grows exponentially with the dimensionality of the robot's configuration space, preventing them from being used in real-time applications. Traditional sampling-based planners such as Rapidly-exploring Random Trees (RRT) [49] and Probabilistic Roadmaps (PRM) [48] overcome this curse of dimensionality by sampling and implicitly constructing a roadmap over the configuration

space. These planners are probabilistically complete; however, without computationally expensive modifications there is no guarantee on optimality nor can they determine if no solution exists [50].

One of the first algorithms in this area was the PRM [48] algorithm, which was designed to be multi-query. Since the original PRM algorithm, a number of modifications have been developed to address different issues, such as reducing the number of nodes needed for the roadmap [102], the ability to find paths through narrow regions [103] and reducing the time spent collision checking [104]. Furthermore, PRMs can be applied to problems with complex constraints such as kinodynamic planning problems [105].

The RRT algorithm [49] is another fundamental sampling based planner. However, unlike the PRM algorithm the graph created is a tree and designed for single query planning. As with PRM, there have been several modifications and improvements to RRT since its conception, such as Single-query Bidirectional Lazy (SBL) planner which applies a lazy collision-checking strategy to reduce the planning time [106]. SBL and RRT-Connect (Bi-directional RRT) [49] grow two trees simultaneously to perform a bidirectional search which further reduces computation time.

While both PRM and RRT both possess theoretical guarantees such as probabilistically completeness, they tend to produce sub-optimal solutions. For RRT this is due to it having no notion of optimality. Rather, the focus is on producing a feasible solution and thus treats all feasible solutions equally. In fact, it was proven that the best paths RRT finds almost surely converges to a non-optimal value [50]. For PRM, the sub-optimality comes from the graph construction which often is designed to achieve coverage of the search space via random sampling. The solution will then be optimal with respect to the constructed graph. There are smoothing and shortcut algorithms [107, 108] that mitigate this; however, they require an additional post processing step. Furthermore, considerable computational effort is spent in sampling and connecting samples in portions of the configuration space that might not be relevant to the task.

Apart from theoretical performance improvements, efforts have been made to speed up the implementation of RRT and PRM methods. KD-trees have been used to speed up neighbour hood search when connecting nodes [109]. Further, parallelising these algorithms and

running multiple searches has proven to be effective at speeding up and improving their performance [110–112].

Any-time planners combine ideas from grid-based search, incremental and sampling-based planners. They are more practical than traditional algorithms because they allow the agent to safely begin executing plans in real-time while during execution they can continue improving the plan.

Anytime Dynamic A* (AD*) [113] is a re-planning algorithm that combines the benefits of anytime and incremental motion planning methods. The algorithm plans a sub optimal path quickly initially and improves on it over time as more information is gathered about the environment. Incremental Phi* is an incremental and more efficient version of Theta* [114].

Optimal, any-time versions of sampling-based planners such as RRT*, PRM* [51, 115] and discretisation-based approaches, R* [116] and ARA* [117], are promising yet are currently computationally inefficient for solving high-dimensional motion planning problems. Informed RRT* is a simple modification to RRT* and provides significantly faster solution convergence for problems where it is not necessary to explore a large portion of the configuration space [118]. FMT* was proven to work well in high-dimensional configuration spaces and provided better solutions than RRT* [119]. It utilises a marching method which orders a search based on cost-to-come; however, this search must be restarted if a higher sampling resolution is needed. RRT$^{\#}$ improves upon the idea in RRT* by propagating local wiring to the rest of the tree and utilises incremental planning techniques to speed up optimality convergence [120].

Batch Informed Trees (BIT*) [121] and its recent improvement Advanced BIT* (ABIT*) [122] balances the benefits of graph search and sampling based planners. They do so by processing batches of samples, as in previous optimal sampling-based planners; however, they additionally update the path between samples efficiently by leveraging incremental search techniques and incorporating new samples into the existing search. BIT* was shown to find better solutions faster than Informed RRT* and FMT* with faster anytime convergence towards the optimum, in particular, for higher-dimensional problems.

The idea of motion planning in task space was utilised in the TS-RRT [87] algorithm, a modification of the RRT algorithm where samples are drawn in the task-space and connected in the configuration space using techniques from task-space control. This results in more efficient planning since sampling is performed in the lower dimensional task space and guides the high-dimensional planning which still occurs in the configuration space in order to satisfy the system's kinematic constraints. However, unlike HAP, this approach does not mitigate potential large changes in joint configurations when planning through the task space.

Luna et al. proposed a sampling based motion planner called XXL which computes high quality trajectories that scale well to high-dimensional robots [123]. While they reported high performance, their method depends on probabilistic sampling and suffers from the limitations of other similar methods. Furthermore, their method is only probabilistically complete with no optimality guarantees.

Recently, Hauer et al. showed that the probability of sampling a point that improves the solution decreases exponentially with the dimension of the problem [124]. In addition, they developed the DRRT algorithm which improved optimality convergence by performing gradient descent on previously sampled points. The convergence rate was shown to remain almost constant for increasing problem dimension. While this is a promising result, when applied to a six-DOF manipulator problem with obstacles it suffered from high computation times and low success rates. This suggests that the algorithm works well in an ideal setting with perfect heuristics and no non-linear costs yet does not generalise well to real-world, constrained problems.

2.2.3 Trajectory Optimisation

Trajectory optimisers have shown great promise in high dimensional systems due to their fast computation and efficient trajectories [125, 126]. Trajectory optimisation is a loose term for methods that produce trajectories through optimisation of some initial, usually random or naive, seed trajectory. This seed is incrementally optimised over time until an optimal or near optimal solution is found. One of the most common approaches is gradient descent [127], which is a first-order optimisation algorithm. In the context of planning a

trajectory for a robotic manipulator, a local minimum is found by incrementally shifting a given trajectory in the direction proportional to the negative of the gradient of a function at a current point.

Gradient Descent has been mainly beneficial for producing smooth, locally optimal trajectories for high-DOF manipulators due to its fast performance and easy implementation; however, without modification it does not take into consideration constraints such as avoiding obstacles. In fact, trajectory optimisation wasn't traditionally used for robot motion planning since obstacles and other constraints requires solving a non-convex, constrained optimisation problem.

Covariant Hamilton Optimisation Motion Planner (CHOMP) [128] by Zucker et al. revived interest in these methods by showing effective real-time performance on several robotic platforms in the presence of obstacles. While CHOMP uses functional gradient techniques to iteratively improve an initial trajectory there exists gradient free versions, such as STOMP [129] and ITOMP [130].

One drawback to CHOMP is that the obstacle costs are based on a signed distance field (SDF) computed over the workspace which can be costly to compute. Schulman et al. proposed an alternative algorithm called TrajOpt [53, 131] which was shown to compute better trajectories quicker and with a higher success rate than CHOMP. TrajOpt differs from CHOMP in that it uses a second-order, sequential convex optimisation procedure called sequential quadratic programming (SQP) and convex-convex collision checking. It is argued that this collision checking method results in better approximation of the configuration space and hence better collision avoidance since it attempts to translate the entire link out of collision, rather than individual points along the link, as with CHOMP. TrajOpt can additionally incorporate various constraints, such as orientation, positional, and even differential constraints.

Gaussian Process Motion Planner 2 (GPMP2) [54, 132] by Mukadam et al. attempted to speed up the trajectory optimisation problem by reducing the amount of collision checking along the trajectory by using probabilistic inference to only check configurations that are likely to be in collision. GPMP2 produced promising results with faster planning times and higher success rates than TrajOpt and CHOMP. However, as with CHOMP, it uses

an SDF to query obstacle costs and suffers the same problem of trying to move individual points along the links out of collision rather than the entire link. Further, the algorithm is difficult to tune due to the large number of parameters.

One major drawback of trajectory optimisers is their susceptibility to local minima. The importance of trajectory initialisation in avoiding these minimas is discussed by Schulman et al. in [53]. A good initialisation has a significant impact on convergence speed and success rate of trajectory optimisers. They further noted that these methods are not guaranteed to find a collision-free solution as the no-collisions constraints in the optimisation problem are non-convex.

A commonly used initial seed is a straight line initialisation in configuration space generated by linearly interpolating between start and goal configurations for a fixed number of points. If this naive initialisation fails then a common method is to try multiple random perturbations of this straight line initialisation. There have been attempts at learning classifiers to predict the effectiveness of a list of trajectory initialisations [133–135]; however, these initialisations are just random perturbations of the naive straight line initialisation. Recently, a sequence-to-sequence learning-based approach was proposed to predict a good initial trajectory conditioned on the environment and task [136]. Their approach uses a encoder-decoder architecture with Long Short Term Memory (LSTM) neural networks and showed better performance than naive straight line initialisations, even in novel environments. Similarly, the library trajectories computed by HAP can be used as good initialisations since they consider the robot structure and prior environment and task information.

An alternative approach to overcoming local minima is to construct a graph over the state-space such that multiple path homotopy classes are considered. This was first considered by Huang et al. in GPMP-GRAPH [137] which builds on the idea of leveraging factor graphs in GPMP2 [54]. Here, an interconnected network of trajectories is evaluated simultaneously instead of a single trajectory chain. This was then extended to run quickly for online settings in POSH [138] and then for dynamic environments in JIST [139].

Regardless of the successes of these state-of-the-art motion planners, there are scenarios where they break down and fail. In this thesis we will demonstrate such scenarios and

motivate the need for a faster, more reliable and efficient method. As will become apparent, HAP accommodates existing motion planners and can be used in conjunction with them to improve their performance.

2.3 Task and Motion Planning

The motion planning methods mentioned above are usually queried on a single start and goal pair. We argue that for many practical applications this is unrealistic and that treating each task in isolation leads to several issues. For multiple tasks, particularly long horizon sequences, the robot can arrive at awkward configurations at a point along the sequence. For example, a manipulator can approach at, or near, joint limits and singularities potentially requiring a large change in configuration to reach subsequent goals in the sequence. Most TAMP approaches tend to focus more on feasibility of the sequence of tasks and specified constraints and tend to disregard the efficiency of the sequence [67, 68, 140–142].

In this thesis the only task level constraints we consider are achieving a given goal pose, remaining within joint limits, and avoiding collisions. As such, other more suitable terms for this problem setting exist in the literature such as task sequencing [69–72] and multi-goal path planning [73, 74]. Several methods have been proposed to solve such problems while considering the kinematic redundancy of the arm, most of which are specialised variants of the travelling salesman problem (TSP) [2]. However, these approaches are computationally expensive and designed for industrial settings where plans can be computed offline since the environment and tasks are fully known beforehand.

To find an efficient sequence in an online setting an accurate heuristic is needed. An accurate heuristic can be computed; however, obtaining this heuristic is non-trivial and often equivalent to solving the original problem [124]. Typically, trajectory costs for manipulators are approximated by the Euclidean workspace distance between the start and goal location of the task [14, 21]. While this is inexpensive to compute it can heavily underestimate the true cost of the trajectory used to execute the task given the non-linearity of high-dimensional manipulators [57], especially when obstacles are present.

In contrast, the library trajectory produced by HAP provides close-to-true costs that take into consideration the kinematic redundancy and non-linear motion model of high-dimensional manipulators. Furthermore, biasing goal configurations to the HAs found by HAP allows for efficient and reliable concatenation of trajectories over long task sequences.

2.4 Coverage Planning

The work in this thesis is related to coverage planning for robot manipulators. In [143, 144] Hassan and Paus et al. proposed methods for optimally assigning a subset of the environment's workspace to robot base poses. While their frameworks factor in trajectory costs and kinematic feasibility, the main objective is to assign non-overlapping regions of the workspace to robot base poses such that the entire workspace is covered. While our method could be used for coverage planning applications, our objective is different in that we are aiming to capture subspaces where the arm can move around within easily which allows for planning to any arbitrary pose within the workspace rather than simply achieving coverage of a discrete set of poses. Additionally, these subspaces are encouraged to overlap which allows for greater flexibility in how tasks can be assigned and need not necessarily be assigned to one specific robot base pose. Furthermore, they assume a known model of the environment which doesn't change in the online scenario, where as our method does allow for this.

2.5 Next-Best-View Planning

Our active perception work is related to next best view (NBV) planning, which attempts to find the best sensor pose, or viewpoint, for reconstructing or inspecting a scene. Reconstruction implies that a model of the scene is not known *a priori*, whereas inspection typically makes use of a prior model. The work in this thesis supports both problem settings; however, we are mainly concerned with the former with the exception of knowing the bounding volume in which the objects of interest reside within. NBV approaches can be further decomposed into several categories, including surface-based [145, 146], search-based [85] and hybrid [147] approaches.

Surface based approaches exploit the geometry of the observed surface [145, 146, 148], where as search based methods perform some search over a volumetric [149] or optimisation [150] based representation of the environment. Surface-based methods are convenient when high quality reconstruction is required; however, they are computationally expensive. Hence, volumetric representations are usually preferred due to their simple visibility operations and probabilistic occupancy estimation [85]. Recently, Kriegel et al. proposed a method that utilises both a mesh and probabilistic volumetric representation to produce a high quality reconstruction while speeding up planning [147].

Recently, a novel density representation was introduced in an attempt to alleviate the complexity of surface based approaches and limited resolution of volumetric approaches [151]. The authors achieve a high resolution reconstruction by operating directly on the fused point cloud; however, their method relies on the sequence of views being close together with only small surface discontinuities. Hence, it cannot be applied effectively to reconstruct sparsely located objects in noisy scenes.

Learning-based approaches have emerged recently. In [60], Hepp et al. use a convolutional neural network (CNN) that takes as input a novel representation of the scene and learns a utility function. While their method outperformed other state of the art utility functions, learning based methods require large datasets to train on and it is unknown how well they generalise to different scenes. For example, it was found that their method had high dependence on the surface voxel distribution.

Another learning-based approach proposes a reinforcement learning solution to the NBV problem [61]. They attempt to learn a policy that exhibits general "look around" behaviour which is agnostic to the task or content in the scene. They utilise a recurrent neural network (RNN) based architecture to learn context about the environment and encode the history of observations to avoid redundant information gain. While their method performed well and appeared to generalise to unseen images, only two datasets were used. Further, they do not account for trajectory costs or environmental and kinematic constraints.

In [85], Delmerico et al. proposed new VI formulations and carried out an extensive comparison with the state of the art, including average entropy [147] and area factor [152]

methods. Their proposed proximity count VI performed comparably with area factor. An interesting result was that within a small number of views, a high level of reconstruction was achieved regardless of the VI formulation.

The area factor's objective is twofold: to perceive unknown areas and to provide some overlap between views to assist with point cloud registration. It achieves this by encouraging views to observe a desired ratio of observed and unobserved information. More specifically the best view observes a desired ratio of occupied and occplane voxels; unknown voxels that are adjacent to unoccupied voxels. While this is reasonable in the case of a full scene reconstruction, it is too restrictive in scenarios where we want to reconstruct ROIs that may be sparsely located. Further, it may be difficult to tune this ratio when there is a large amount of occplanes in the scene due to noisy clutter.

In [152], Vasquez-Gomez et al. utilised a quality factor, as formulated by Lozano et al. in [153], where views that place the sensor orthogonal to the surface are favoured. While their quality measure is applied to any surface in the scene, our work only considers the measure for ROIs and uses it as an activation parameter rather than a continuous scalar factor. Our work further motivates orthogonal views in order to avoid occlusion and encourage overlap, something that is addressed separately by the area factor formulation. Vasques-Gomez additionally proposed a decaying sensor resolution reduction technique to increase computation speed. This work uses a similar approach; however, instead of decaying the reduction, the original point cloud used to construct the volumetric representation is retained and leveraged for richer information.

Vasque-Gomez recently introduced a slight improvement to the area factor formulation called frustum information gain (FIG) [154]. This formulation sums the entropy of occluded unknown voxels, called the sensor frustrum, observed from a view. Unlike Kriegel's average entropy, the FIG formulation avoids multiple IG contribution from the same voxels. Our work implements a similar idea and deals with synchronising this across multiple robots and over a finite planning horizon.

The use of point cloud segmentation for active scene understanding has been considered in previous work. Recent segmentation algorithms have enabled robust classification of partially observed objects in cluttered environments [75, 155–157]; we leverage Gaussian

process implicit surface (GPIS) segmentation by Martens et al. [155] to classify ROIs. The GPIS approach "hallucinates" unseen parts of the scene to make predictions. Related prediction models that could potentially be used within our framework include [77, 158, 159].

Authors in [160] propose a method for generating candidate views dynamically by assigning them to frontier cells in the scene. Views that observe objects with higher saliency are given higher reward. The same authors later proposed a surfel-based representation for increased computation speed and allowing for larger scenes to be reconstructed [159].

Commonly, the focus of the aforementioned NBV methods is on achieving a complete, high-resolution reconstruction of a scene. These methods generally do not work well when the objects are sparsely located or when only certain areas are of interest, which we aim to address. Many NBV approaches perseverate on found objects while ignoring the potential information gain of exploring the environment to observe unseen objects. While exploration is often incentivised in map building scenarios, it is rarely combined with other objectives. Best et al. [161] define reward regions for both observed object parts and unexplored space; we propose a similar technique that is suitable for cluttered environments with occluded ROIs.

2.6 Information Gathering

Prior work for NBV planning tends to focus on perception prediction models and considers only simple, myopic planning algorithms. Conversely, work in informative path planning presents sophisticated planners [19, 20, 22], sometimes with performance guarantees, with relatively-simple or loosely-specified perception objectives. In both cases, approaches often do not consider trajectory costs due to kinematic or environmental constraints. Monte Carlo tree search (MCTS) [162] methods can potentially overcome these limitations, enabling non-myopic planning with respect to complex perception objectives and motion constraints, while providing convergence guarantees [82, 163–166].

2.7 Multi-Robot Active Perception

NBV and related information gathering problems have typically only been formulated for single robots; few approaches have been scaled up for multi-robot systems, particularly in decentralised settings. Market-based approaches [167] are commonly used for decentralised planning in other contexts, yet the formulations are typically not expressive enough to address challenges such as considering dependencies between observations. The Dec-POMDP is a more expressive formulation [168], yet the solution approaches are generally offline and centralised. The recently-proposed decentralised Monte Carlo tree search (Dec-MCTS) algorithm [82] by Best et al. generalised MCTS for decentralised multi-robot planning, and is an anytime and online algorithm that has been demonstrated to work well for complex perception tasks. We use Dec-MCTS in our multi-robot active perception approach and provide the first demonstration of how it can be combined with a low-level motion planner to plan for a team of robots that have high-dimensional configuration spaces.

Few of the methods referenced above have been demonstrated to work with real robots. In this work we evaluate our methods in hardware with two robot arms in complex scenes with occluded ROIs, significant sensor noise and high-DOF robotic arms.

Chapter 3

Hausdorff Approximation Planner (HAP)

This chapter introduces HAP and provides implementation details, practical considerations and how to leverage the HAs generated by the algorithm for online planning. The work here addresses the problem of robust planning for high-dimensional manipulators over long task horizons. Empirical results are presented for several simulated and real hardware experiments with various manipulator platforms.

3.1 Overview

We consider operational scenarios where a series of motion plans with differing goal configurations must be computed in a given environment. The environment consists of dynamic elements such as humans operating in the workspace and objects to be manipulated, and static elements such fixtures and equipment.

Classical approaches such as the probabilistic road map (PRM) [48] aim to gain efficiency through computational reuse; a computationally costly offline process generates a data structure that can then be repeatedly queried efficiently by a low-cost online process. However, this approach can quickly become unwieldy in practice for robotic arms, which often have six or more degrees of freedom. It is necessary to resolve the trade-off between

the size of the precomputed data structure, which grows with the complexity of the environment and desired range of motion of the arm, and the speed of online queries, whose complexity grows proportionally.

One way to limit the expanse of roadmap-like data structures, and thus improve query time, is through the notion of *task space*. Unlike configuration space and work space, which are defined by the physical design of the robot system, task space is defined by a user according to the tasks at hand [78]. An intuitive example is to define a planar task space for operations on a flat surface such as sanding or linishing. For pick-and-place operations, the task space can be naturally expressed as the set of poses for possible object locations. A set of trajectories can be generated that correspond to common start-goal locations, for example, and reused online [63, 64]. These are known as library trajectory methods and can significantly improve online plan computation times. However, current methods tend to not consider trajectory cost optimisation in the library construction which can lead to arbitrary performance when adapted online. Instead HAP introduces structure into the library which leads to more predictable and well-behaved execution online, particularly over long task horizons.

A desirable property of motion planning in task space is that, for any two tasks close to each other in task space, one expects a smooth, short trajectory to exist between them. The key idea we propose is to encourage such behaviour through the trajectory library construction. We achieve this by establishing a bounding relation between distances in task space and corresponding distances in configuration space, defined as a mapping between metric spaces known as an ϵ-Gromov-Hausdorff approximation (ϵ-GHA), or Hausdorff approximation (HA) for short. The aim is to ensure that a short path in task space implies a short path in configuration space, which additionally encourages smoothness.

We present HAP, a novel algorithm for efficient and reliable motion planning in task space. HAP builds a set of precomputed trajectories that resemble roadmap methods in that trajectories can be concatenated to satisfy a sequence of queries. To allow concatenation, the precomputed trajectories are indexed by a set of maps from the task space (end-effector poses) to the configuration space (joint angles) that disambiguate kinematic redundancy. The maps are designed to be HAs, which are approximate isometries that, as we show,

ensure that the trajectories are efficient in combination. As a result each trajectory is efficient individually since the task space is divided into smaller disjoint subspaces of similar manipulator configurations and requiring that, for any two poses close to each other in a given subspace, there exists a short trajectory between resulting configurations according to the HA mapping.

Given a user-defined task space, HAP automatically computes a subspace decomposition and generates a set of trajectories that span it. These trajectories are stored in a library that can be retrieved and modified online to adapt to changes in the environment and additionally to changes in the task space definition.

In this chapter we evaluate the behaviour of the algorithm and demonstrate its applicability to a variety of manipulator platforms and task space definitions through experiments in simulation with a six-DOF, with and without a mobile base, and a seven-DOF manipulator. Results show that the extra kinematic redundancy afforded in the seven-DOF cases leads to simpler subspace decompositions with fewer divisions than in the six-DOF case, which agrees with our intuition. Planning time performance is favourable in both cases, however, which shows how the HAP algorithm is effective in choosing precomputed trajectories for rapid online planning even in the presence of physical constraints, such as problematic singularities and joint limits, that otherwise would lead to unnecessarily long or awkward trajectories. This additionally leads to much higher planning success rates. Results for the mobile manipulator case show how the notion of subspaces naturally generalises to more complicated kinematics.

The main significance of this work is to improve the practical utility of autonomous robotic manipulators, particularly for non-experts. Automatically generating smooth, short trajectories online directly from a user-defined task space with reasonable performance guarantees is a step beyond current general purpose motion planning methods, which generally require expertise to design implementations that perform well in practice.

3.1.1 Chapter Outline

The remainder of this chapter is organised as follows. Section 3.2 formally defines the task and motion planning problem and how HAP reduces this problem into separate motion planning and task sequencing problems without loss of expressiveness. Section 3.3 explains how HAs and the trajectory library generated by HAP are used for online planning. Section 3.4 details how the HAs are computed and then how a trajectory library is generated from them. Section 3.6 provides a time complexity analysis of the trajectory library construction algorithm and a bounded suboptimality analysis. Section 3.7 evaluates HAP on several different problem settings and robotic manipulator platforms. A robotic pruning experiment is presented in Section 3.8 which demonstrates the applicability of HAP to a real system.

3.2 Problem Formulation

3.2.1 Notation

C represents the configuration space. Given a configuration $q \in C$, $A(q) \subset W$ denotes the space occupied by the robot model at configuration q. $W_{env} \subset \mathbb{R}^3$ is the model of environmental obstacles, for example a general bookshelf structure. The obstacle region is defined as $C_{obs} = \{q \in C \mid A(q) \cap W_{env} \neq \emptyset\}$, from which we obtain the free space region $C_{free} = C \setminus C_{obs}$. The task space $\mathcal{T} \subset SE(3)$ is the set of poses of the robot's end effector. $T \subset \mathcal{T}$ represents a finite set of online tasks assigned to the robot. $\hat{T}$ is a discrete approximation of the subset of $\mathcal{T}$ where the robot is expected to operate frequently.

3.2.2 Task and Motion Planning Problem

The manipulator is given a set of tasks, modelled as a finite set of 6D poses $T = \{t_1, ..., t_M\} \subset SE(3)$. To complete the tasks, the manipulator operates repeatedly within a particular subset of the task space. The environment is not entirely known in advance; however, general models are available that approximate what is expected.

The manipulator's trajectory is modelled as a discrete sequence of configurations $\pi = \{q_1, ..., q_N\}$. We measure the length of a trajectory using a metric on the configuration space:

$$d_C(\pi) = \sum_{i=1}^{N-1} d_C(q_i, q_{i+1}). \tag{3.1}$$

We are interested in finding the shortest collision-free trajectory that completes all tasks in T.

Problem 3.1 (Task and motion planning problem)**.** *Specifically, the task and motion planning problem can be written as:*

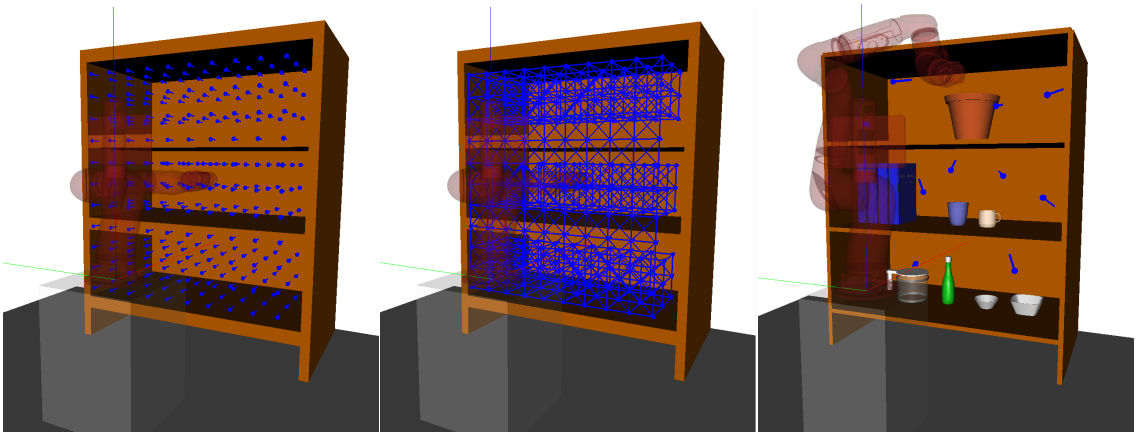
$$\begin{aligned} \min_{\pi \in C_{free}^N} \quad & d_C(\pi), \\ s.t. \quad & \forall t \in T, \pi \cap Q(t) \neq \emptyset. \end{aligned} \tag{3.2}$$

It is challenging to solve (3.2) directly. For each task pose t_i to be visited there is a set of valid configurations $Q(t_i)$, and a direct solution requires simultaneously sequencing the tasks while choosing the optimal configuration for each task. In other words, it contains the set-TSP, which is harder than the standard TSP [169] and potentially too computationally expensive for real-time planning. The number of possible sequences is order $O(|Q(t)|^{|T|})$ where $|Q(t)|$ is the cardinality of the largest set of IK solutions for a single task $t \in T$, and $|T|$ is the number of tasks.

3.2.3 Problem Reduction

To address the above complexities, we reduce the set-TSP to a classical TSP by mapping each task pose t_i to a suitable unique IK solution $\theta(t_i) \in Q(t_i)$. The map $\theta : \hat{T} \to C$ is chosen to approximate an isometry, as we detail in Section 3.4. Intuitively, an approximate isometry enforces two poses t_i, t_j that are close in task space to remain close in configuration space after mapping $\theta(t_i), \theta(t_j)$.

The task and motion planning problem then decomposes into a motion planning problem and a task sequencing problem that can be solved separately without loss of expressiveness:



(a) (b) (c)

Figure 3.1: Example environment and task space next to an anticipated online scenario: (a) shows the discretised task space $\hat{T}$ (blue poses) and inflated environment m, (b) shows an undirected graph G constructed over discretised task space $\hat{T}$, where nodes are tasks in $\hat{T}$, (c) is an example online scenario with the arm in its home configuration. Online, HAP is robust against objects in (c) that are unmodelled in m and tasks can differ to those in $\hat{T}$.

Problem 3.2 (Motion planning problem)**.** *The motion planning problem is to find the shortest path in the configuration space between each task poses t_i and t_j:*

$$d^*(t_i, t_j) = \min_{\pi} d_C(\pi). \tag{3.3}$$

Problem 3.3 (Task sequencing problem)**.** *The task sequencing problem is to find a permutation of the tasks $\sigma \in S_M$ that minimizes the total cost (i.e. a classical TSP):*

$$\min_{\sigma \in S_M} \sum_{i=1}^{N-1} d^*(t_{\sigma(i)}, t_{\sigma(i+1)}). \tag{3.4}$$

Note that for some task spaces, a single map with the approximate isometry property may not exist. In this case, K maps $\theta^i : \hat{T}^i \to C^i$, $i = 1, \ldots, K$ with the required property can be found between covering sets of subspaces of task and configuration space. Then, the above problems must be solved independently for each subspace.

Algorithm 3.1 Online Planner

1: Given an online environment and tasks, match with candidate library trajectories (Sec. 3.3.1)
2: Modify trajectories to satisfy tasks (Sec. 3.3.2)
3: Compute optimal sequences using TSP solver with computed edge costs (Sec. 3.3.3)
4: Adapt and process modified library trajectories for execution within and across subspaces (Sec. 3.3.4)

3.3 Online Motion Planning with HAs

In this section, we outline the online planning algorithm that solves the motion planning problem (3.3) and task sequencing problem (3.4), as summarised in Algorithm 3.1.

3.3.1 Retrieval of Trajectories

To generate a plan that completes the set of online tasks T, we first match the tasks to those modelled in the library $\hat{T}$, and retrieve the corresponding trajectories. We begin with the case of a single map θ, where there is no division of $\hat{T}$ into subspaces.

Each online task $t \in T$ is first associated with the k-closest tasks in the library, $\hat{t}^k \in \hat{T}$ in terms of the task-space distance d_T. We then compare all IK solutions $q_t \in Q(t)$ of the online task to the unique IK solutions $\hat{q}_t^k = \theta(\hat{t}^k)$ of the k-closest task poses and pick the pair of IK solutions $(q_t^*, \hat{q}_t^*)$ that minimize a similarity metric, for example this metric could simply be the L_2-norm. Then, for all pairs of online tasks $(t_i, t_j) \in T$, the precomputed trajectory between matched library tasks $\hat{\pi}_{\hat{q}_{t_i}^* : \hat{q}_{t_j}^*} = \{\hat{q}_{t_i}^*, \ldots, \hat{q}_{t_j}^*\}$ is retrieved.

The reason for using both task and configuration space distances, as opposed to configuration space distance alone (as in, e.g., [63, 64]), is as follows. While configuration space distance allows finding the IK solution that causes the least joint movement, task space distance restricts the match to reflect the workspace geometry. Using the bookshelf environment in Figure 3.1 as an example, an online task may be located in the middle shelf yet be matched with a library trajectory whose end point is located in the top shelf due to a smaller configuration difference, causing a collision when the IK solution for the online task is appended. Meanwhile, a library trajectory whose end point is matched using task space distance is more likely to lie in the middle section of the bookshelf, accounting for

the shelf geometry. Further, task-space comparison allows rough initial matching when sequencing multiple tasks as the configuration of the robot after the first task is unknown prior to planning. Note that the use of k-closest task poses allow for more flexibility when matching in terms of configuration space distance.

The process is similar for the multiple map θ^i case; however, two online tasks assigned to different subspaces will not have retrievable library trajectories. To handle this, online tasks are grouped based on their assigned HA and trajectories are only retrieved between tasks in the same group.

3.3.2 Modifying Trajectories to Satisfy Tasks

With the library trajectory $\hat{\pi}_{\hat{q}^*_{t_i}:\hat{q}^*_{t_j}}$ retrieved, the online configurations $q^*_{t_i}$ and $q^*_{t_j}$ are appended to the beginning and end to obtain the modified trajectory $\pi_{q_{t_i}:q_{t_j}} = \{q^*_{t_i}, \hat{q}^*_{t_i}, \ldots, \hat{q}^*_{t_j}, q^*_{t_j}\}$ which satisfies the given online tasks. Provided a good coverage of the workspace in the trajectory library generation, there should exist an IK solution for the online tasks that matches closely with the end points of the library trajectory. After modification, the trajectory cost $d_C(\pi_{q_{t_i}:q_{t_j}})$ is computed.

If a pair of tasks belong to different HAs, there is no corresponding trajectory in the library. In this case, the corresponding trajectory is regarded as a straight line to and from a designated home configuration that is chosen to minimise excessive travel (see Section 3.4.3.4). While one could use a motion planner to compute trajectories between the home configuration and all library configurations, using a straight line was found to be sufficient in practice because we ensure the home configuration is sufficiently far away from obstacles. Additionally, using a simple straight line prior allows for flexibility in choosing the home pose in the online scenario if necessary.

For multiple HAs, finding a best match for each task would require performing the above for every map. Depending on the number of HAs, tasks, manipulator DOFs and environment complexity this can add significant overhead computation. To reduce the computation required for each online task, the trajectory library's task subspaces are first sorted by their proximity to the task. The sorted subspaces are then iterated over and the one

with the closest task, using the similarity metric, to any of the online task's IK solutions is assigned. The search over subspaces is terminated when a matched IK solution has a similarity difference less than some threshold. Doing so should reduce the amount of iterating over subspaces and reduce computation; however, if necessary one could easily parallelise the process since the retrieval of trajectories for each HA is independent of each other.

3.3.3 Task Sequencing

The modified library trajectories between each pair of tasks are sequenced using a TSP solver. A weighted adjacency matrix is formed for each subspace group based on the trajectory costs computed using the modified library trajectories. A TSP solver then generates the optimal task sequence based on the weighted adjacency matrix.

For multiple maps, a TSP is independently solved for tasks within each subspace. The sequence in which subspace tasks are carried out is simply the order in which they were found during the library search. For the mobile robot case we ignore the base movement cost when switching between subspaces; however, if this were important then sequencing the group execution order could be solved using another TSP with the base movement costs. For both the single and multiple map case the fixed home configuration is assigned to the depot in the sequence.

3.3.4 Online Adaptation of Trajectories

After sequencing the modified library trajectories, the trajectories must be post-processed to ensure that they are time-continuous safe and collision-free. This is important because the online scenario may contain obstacles that were not accounted for during the library generation. Therefore, we adapt the sequence of modified library trajectories using off-the-shelf trajectory optimisers and sampling-based planners.

The modified library trajectories are used as initial seeds in a trajectory optimiser first. Planners such as TrajOpt [53] can move trajectories out of collision and guarantee time

continuous safety. If the trajectory optimiser fails then a probabilistically complete planner, such as BIT* [121], can optionally be used as a backup to plan the trajectory. While no prior trajectory is incorporated here, using the end points of the modified trajectories as goal configurations is helpful and acts as a heuristic for the probabilistic planners to bias their sampling towards.

3.4 Trajectory Library Construction

In this section we detail how the HAs are computed given a robot manipulator, possibly mobile, and an anticipated environment and task space. From these HAs the trajectory library is generated by simply taking shortest paths through the discretised task space and then mapping them to configuration space. Additionally, practical considerations are given which describe implementation specific details to aid practitioners when applying HAP.

3.4.1 Algorithm Overview

To construct the trajectory library used for online planning, we first anticipate an environment m and a set of tasks $\hat{T}$ that are representative of online scenarios. An undirected graph $\mathbb{G}$ is created with nodes corresponding to tasks $t \in \hat{T}$ and edges formed via a max connection radius (see Section 3.4.3.7). Then, Algorithm 3.2 generates a trajectory library using the undirected graph $\mathbb{G}$.

Algorithm 3.2 is initialised by assigning all nodes to $\hat{T}_{\text{open}}$. A node stays in $\hat{T}_{\text{open}}$ until a unique IK solution is assigned. While $\hat{T}_{\text{open}}$ is not empty, θ is searched for via a modified Dijkstra procedure (Algorithm 3.3).

Algorithm 3.3 searches for a θ that minimizes the objective cost: the sum of all minimum cost paths $\Gamma^*_{t_0:t}$ from a root node $t_0 \in \mathbb{G}$ to all other nodes. That is,

$$J(\theta, t_0) = \sum_{t \in \mathbb{G} \setminus t_0} g(\Gamma^*_{t_0:t}; \theta), \tag{3.5}$$

where the cost of any path $\Gamma_{t_0:t_{N-1}} = \{t_0, \ldots, t_{N-1}\}$ of N nodes is defined as

$$g(\Gamma_{t_0:t_{N-1}}; \theta) = \sum_{l=0}^{N-2} d_C(\theta(t_l), \theta(t_{l+1})). \tag{3.6}$$

To ensure that trajectories have bounded lengths and do not involve large, unnecessary arm movements, θ is additionally constrained such that it is an approximate isometry, or an ϵ-Gromov-Hausdorff approximation (ϵ-GHA). Given a distance metric in configuration space, d_C, and task space, d_T, an ϵ-GHA is defined as:

Definition 3.1 (ϵ-Gromov-Hausdorff approximation (ϵ-GHA))**.** *The map* $\theta : (\hat{T}, d_T) \rightarrow (C, d_C)$ *is an* ϵ*-Gromov-Hausdorff approximation if* $\forall t_1, t_2 \in \hat{T}$

$$|d_T(t_1, t_2) - d_C(\theta(t_1), \theta(t_2))| < \epsilon \tag{3.7}$$

for some $\epsilon > 0$*.*

Depending on the value of ϵ and topology of m, some nodes may still have undefined mappings after a single iteration of the algorithm. It is then necessary to search for multiple maps $\{\theta_i\}_{i=1}^K$ that map a covering set of subspaces $\{\hat{T}_i\}_{i=1}^K \in \hat{T}$ to a set of disjoint subspaces $\{C_i\}_{i=1}^K \in C$. In this case, the routine runs K times to find multiple maps θ^i and subspaces $\hat{T}^i$. The algorithm samples a number of root nodes from $\hat{T}_{\text{open}}$ to the same end. The map(s) with smallest $J(\theta, t_0)$ is selected.

After θ or $\{\theta^i\}_{i=1}^K$ is found, a vanilla Dijkstra algorithm is run on the graph with the resulting fixed, unique IK solutions $q_t = \theta^i(t)$ for each $t \in \mathbb{G}$ and minimum cost paths $\Gamma^*_{t_i:t_j}$ are generated pairwise for all tasks. The minimum cost paths $\Gamma^*_{t_i:t_j}$ are then converted into configurations space trajectories for storage in the trajectory library, $\hat{\pi}$, by applying $\theta^i(\Gamma^*_{t_i:t_j})$.

The library construction algorithm is in a sense similar to region growing methods for segmenting images [170] and 3D point clouds [171] where candidate HAs are grown out from several configuration "seeds" in the graph. The main difference here being that the criteria for set expansion is based on the ϵ-GHA condition in (3.7).

3.4.2 Modified Dijkstra Algorithm for Finding HAs

The *Modified Dijkstra* algorithm as outlined in Algorithm 3.3 attempts to find a unique IK solution for each task in $\hat{T}$ such that the objective cost in (3.5) is minimised. It begins by assigning an undefined mapping to all nodes and mapping a root node to an arbitrary IK solution. The rest of the procedure is carried out as in vanilla Dijkstra with a priority queue $\mathbb{Q}$, with modifications to the node expansion step. Here, the set of neighbouring nodes of t, $\mathcal{N}_t$, is run through the functions shown in Algorithms 3.4 and 3.5.

In *Get Candidate*, if a node $u \in \mathcal{N}_t$ has already been assigned an IK solution q_u, this solution and the edge cost,

$$l(u,t) = d_C(\theta(u), \theta(t)), \tag{3.8}$$

are returned and the map θ is not updated. Otherwise, the IK solution that gives the minimum $l(u,t)$ while satisfying the ϵ-GHA constraint in (3.7) is returned. The returned q_u and $l(u,t)$ are passed as a candidate to *Update*. In *Update*, if the candidate path cost from root node to $u \in \mathcal{N}_t$ is less than the current cost then q_u and the path cost are updated. Additionally, if u is not in $\mathbb{Q}$, it is added.

The above process is repeated for all IK solutions of the root node. If required, the algorithm is run K times to find multiple maps $\{\theta^i\}_{i=1}^K$. The resulting map(s) that minimise $J(\theta^i, t_0)$ is returned. All nodes are assigned an IK solution before being placed in the queue. Additionally, it is important to note that these assignments do not change during an iteration of the routine, as this could result in unstable behaviour and violation of the ϵ-GHA condition due to changing edge costs. However, path costs may change after finding shorter paths through $\mathbb{G}$.

In contrast to vanilla Dijkstra, all nodes are initialised with a non-infinite path cost c_{max}. This way, a candidate IK assignment is only accepted if it does not lead to high cost. Additionally, HAs with greater coverage will be favoured, particularly in earlier iterations.

Algorithm 3.2 Generate Trajectory Library

Input: environment $\boldsymbol{m}$ and poses $\hat{\boldsymbol{T}}$
Output: trajectory library $\hat{\boldsymbol{\pi}}$

$\mathbb{G} \leftarrow$ `gen_edges`$(\hat{T})$
$\hat{T}_{\text{open}} \leftarrow \hat{T}$
$i \leftarrow 0$
while $\hat{T}_{\text{open}} \neq \emptyset$ **do**
  $\hat{T}_{root} \leftarrow$ `sample`$(\hat{T}_{\text{open}})$
  $J(\theta^i, \hat{T}_{root}) \leftarrow \infty$
  for each t_0 in $\hat{T}_{root}$ **do**
    $J(\theta', t_0), \theta' \leftarrow$ `modified_dijkstra`$(\mathbb{G}, t_0)$
    if $J(\theta', t_0) < J(\theta^i, \hat{T}_{root})$ **then**
      $J(\theta^i, \hat{T}_{root}) \leftarrow J(\theta', t_0)$
      $\theta^i \leftarrow \theta'$
    end if
  end for
  for each t in $\mathbb{G} \mid \theta^i(t)$ not UNDEFINED **do**
    $\hat{T}_{\text{open}} \leftarrow \hat{T}_{\text{open}} \setminus \{t\}$
    $\Gamma_{t:-} \leftarrow$ `vanilla_dijkstra`$(\mathbb{G}, t, \theta^i)$
    $\hat{\pi}^i(\theta^i(t), \theta^i(-)) \leftarrow \theta^i(\Gamma_{t:-})$
  end for
  $i \leftarrow i + 1$
end while

3.4.3 Practical Considerations

The following section details practical considerations for implementing HAP on various robot manipulator platforms. This includes a method for encouraging subspace exploration, adding task-based costs to the HAP optimisation objective, modifications for using a mobile base, ensuring smoothing motions when moving between different subspaces and design of the environment and task-space.

3.4.3.1 Encouraging exploration

While searching for θ^i, it is beneficial to bias the search toward the unexplored region of the task space. To encourage subspace exploration in subsequent iterations of the routine, a penalty $\rho \cdot \omega(t)$ may be added to all edge costs passing through a node, where $\omega(t)$ is a count of how many times a node has been assigned to a mapping, and ρ is a tuning parameter. As the penalties $\rho \cdot \omega(t)$ accumulate, previously mapped portions of the graph

Algorithm 3.3 Modified Dijkstra

Input: graph $\mathbb{G}$, root node $\boldsymbol{t_0}$
Output: minimum sum of path costs $\boldsymbol{J(\theta', t_0)}$ and corresponding mapping $\boldsymbol{\theta'}$

$J(\theta', t_0) \leftarrow \infty$
for each q in $Q(t_0)$ **do**
  for each t in $\mathbb{G}$ **do**
    $\theta(t) \leftarrow q$ if $t = t_0$, else UNDEFINED
    $g(\Gamma_{t_0:t_0}; \theta) \leftarrow 0$ if $t = t_0$, else c_{max}
  end for
  while $\mathbb{Q} \neq \emptyset$ **do**
    $t \leftarrow \text{argmin}_{t' \in \mathbb{Q}} g(\Gamma_{t_0:t'}; \theta)$
    $q_t \leftarrow \theta(t)$
    for each u in $\mathcal{N}_t$ **do**
      if $\theta(u)$ UNDEFINED **then**
        $q_u, l(u,t) \leftarrow$ `get_candidate`(q_t)
      else
        $q_u \leftarrow \theta(u)$
        $l(u,t) \leftarrow d_C(q_t, \theta(u))$
      end if
      $g(\Gamma_{t_0:u}; \theta), \theta \leftarrow$ `update`$(\mathbb{Q}, q_u, l(u,t))$
    end for
  end while
  $J(\theta, t_0) = \sum_{t \in \mathbb{G} \setminus t_0} g(\Gamma_{t_0:t}; \theta)$
  if $J(\theta, t_0) < J(\theta', t_0)$ **then**
    $J(\theta', t_0) \leftarrow J(\theta, t_0)$
    $\theta' \leftarrow \theta$
  end if
end for

Algorithm 3.4 Get Candidate

Input: IK assignment of the currently expanded node $\boldsymbol{q_t}$
Output: candidate IK assignment $\boldsymbol{q_u}$ and resulting edge cost $\boldsymbol{l(u,t)}$

$l(u,t) \leftarrow \infty$
for each p in $Q(u) \mid d_C(q_t, p) < \epsilon + d_T(u,t)$ **do**
  if $d_C(q_t, p) < l(u,t)$ **then**
    $q_u \leftarrow p$
    $l(u,t) \leftarrow d_C(q_t, q_u)$
  end if
end for

Algorithm 3.5 Update

Input: Queue $\mathbb{Q}$, candidate IK assignment $\boldsymbol{q_u}$ and edge cost $\boldsymbol{l(u,t)}$
Output: updated path cost $\boldsymbol{g(\Gamma_{t_0:u};\theta)}$ and updated IK assignment $\boldsymbol{\theta(u)}$

1: **if** $l(u,t) + g(\Gamma_{t_0:t};\theta) < g(\Gamma_{t_0:u};\theta)$ **then**
2: $\quad\theta(u) \leftarrow q_u$
3: $\quad g(\Gamma_{t_0:u};\theta) \leftarrow l(u,t) + g(\Gamma_{t_0:t};\theta)$
4: $\quad\mathbb{Q} \leftarrow \mathbb{Q} \cup \{u\}$
5: **end if**

are not considered in later iterations of the algorithm as their path costs may exceed c_{max}. The algorithm then focuses on previously unmapped nodes to increase coverage of $\hat{T}$.

3.4.3.2 Adding a task-based cost

In (3.3) and (3.4), we defined the motion planning and task sequencing objective in terms of trajectory length according to d_C, a metric on the configuration space. We thus assigned IK solutions to nodes in $\mathbb{G}$ via the modified Dijkstra method, using edge costs as in (3.8). However, edge costs in $\mathbb{G}$ are not limited to this form. In fact, one can introduce a task-specific component $c(\theta(u), \theta(t))$ to the edge cost while maintaining bounded trajectory lengths. For example, a task-specific component may be introduced to penalise configurations near joint limits. Then, the map θ found will ensure both a bounded trajectory length and task-specific requirements are met. It is important to note that while the edge costs may be different to d_C the ϵ-GHA condition is still checked using the original d_C metric.

3.4.3.3 Modifications for mobile bases

Greater coverage of the task space with fewer HAs can potentially be achieved by allowing the arm to be mobile. The idea of multiple HAs synergises well with a mobile manipulator. Instead of allocating an arbitrary number of HAs, one can choose a discrete number of base poses and assign HAs to each base pose. This further provides more flexibility in how the arm can allocate poses to HAs.

The optimisation routine for a mobile base is similar to the static base case; however, for each iteration in Algorithm 3.2 the modified Dijkstra optimisation routine is run for all base

poses and the one that yields the lowest objective cost is allocated a HA mapping. This base pose is then removed as a candidate in the subsequent iterations. This is repeated until all base poses have been assigned a subspace or until $\hat{T}_{\text{open}} = \emptyset$. All base poses are considered before allocating any mappings in order to find the base poses that give greater coverage early and lower objective costs. This way, base poses better suited for difficult to reach regions of the task space are reserved for later iterations.

3.4.3.4 Ensuring smooth transitions between subspaces

As previously mentioned, when moving between subspaces the arm first moves back to a fixed home configuration. Based on a given home pose, the corresponding IK solution that minimises the distance to the average configuration of the first HA, q^0_{avg}, is chosen. Explicitly, q^0_{avg} is defined as:

$$q^0_{avg} = \frac{1}{|Im(\theta^0)|} \sum_{q \in Im(\theta^0)} q. \tag{3.9}$$

Note that the home configuration can be computed online allowing for flexible home pose choice. To avoid large changes in configuration while returning to home configuration, the subspaces $\{C^i\}_{i=1}^K$ are biased to be close to one other.

For $i \geq 1$ in Algorithm 3.2 an additional penalty $\rho_s \cdot |\theta^i(u) - q^0_{avg}|$ with user-defined weighting ρ_s is added to the edge cost in (3.8). This way, chosen IK solutions are biased to be close in configuration space to q^0_{avg}. In addition, one can optionally enforce that that the IK solution assignment for the root node t_0 is within some distance threshold, that is $|\theta^i(t_0) - q^0_{avg}| < \zeta$.

3.4.3.5 Limiting the number of subspaces

In Algorithm 3.2 the optimisation routine terminates only when $\hat{T}_{\text{open}} = \emptyset$. However, in some cases this condition may never be met or the number of subspaces required may be large. The former condition can occur if a conservative ζ is chosen; that is, mapping a part

of the task space necessitates a large change in manipulator configuration when switching subspaces and ζ is blocking this from happening. The latter condition is undesirable because it can slow down the trajectory library matching during online planning. Furthermore, frequent subspace switching adds undesirable overhead to the online execution. Hence in practice, the main loop in Algorithm 3.2 can be terminated once a maximum number of subspaces have been found or until the ζ threshold cannot be satisfied. Alternatively, the loop can be terminated once a certain coverage percentage has been achieved or when the size of the subspace found in a new iteration falls below a set threshold.

3.4.3.6 Environment design

An environment representation can be given at trajectory library construction time to inform HA assignment. Obstacles can be represented as a union of basic 3D shape primitives. For example, a bookshelf can be represented as a union of boxes. The shape and size is chosen such that a bounded volume is formed around static parts of the scene. In Figure 3.1 an example of a bookshelf environment is modeled. In a real scenario the geometry of this model should remain roughly the same with only new objects appearing on shelves. To account for sensor noise, modelling inaccuracy and/or localisation error, in the online scenario the bookshelf is inflated slightly. Inflating the obstacles additionally encourages library trajectories to have greater clearance from the obstacles, increasing robustness to new unseen obstacles and allowing more room for trajectories to adapt and move around them.

In this thesis obstacle scenes are defined manually for a range of obstacle configurations; however, a strategy could be developed to create these procedurally. Further, although not in the scope of this thesis, one could implement a learning algorithm to choose a trajectory library based on the obstacle configuration given in an online setting. Similar strategies have been used in previous work [133–135].

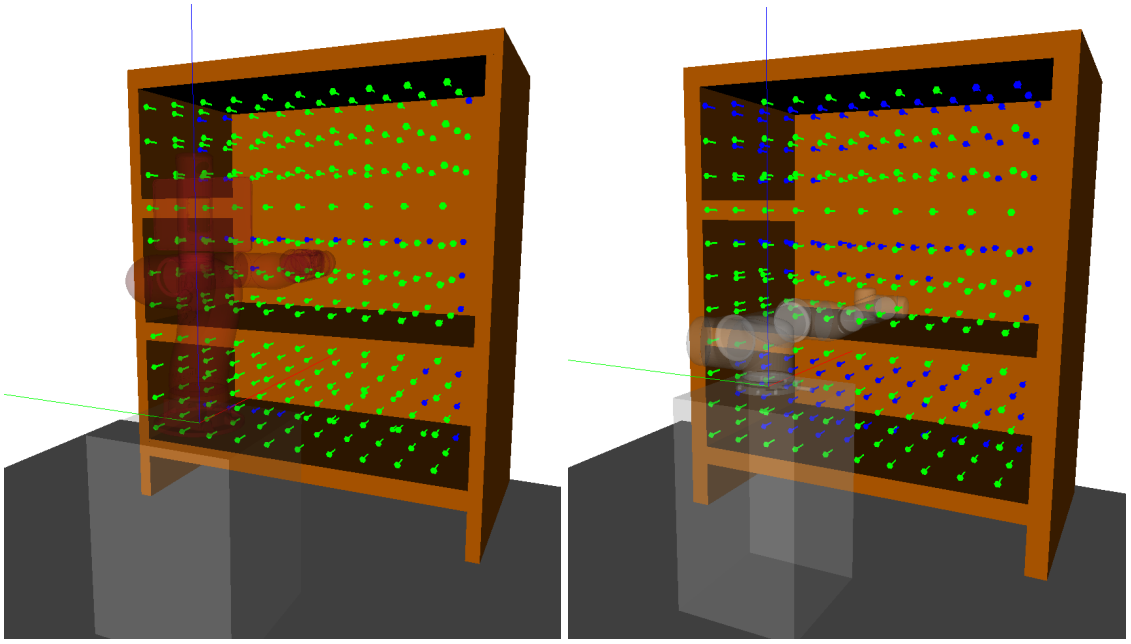
3.4.3.7 Task space graph construction

The chosen $\hat{T}$ is a discretisation of the user-defined task space. In the context of this work it represents approximate poses that the arm is expected to plan to. An example discretisation strategy and graph construction is visualised in Figure 3.1a. The poses here could represent abstract tasks such as pre-grasp points or camera viewpoints for active perception. Notice that in the online scenario in Figure 3.1c the task poses need not lie exactly on the graph, however, performance may decrease the greater this discrepancy is.

Nodes can be defined in any way; however, the intuition is that greater edge connection density allows for more diverse paths and greater node density increases the chance of time continuous safety being met by the trajectory library. The orientation of the poses in the graph should align roughly with the expected tasks. For example, in Figure 3.1a the poses are all pointing forward into the bookshelf which is suitable for tasks such as grasping and scene reconstruction.

Similarly, the edge connection strategy is flexible; however, one should consider that the edges are connecting nodes in the workspace and hence a local connection strategy needs to be devised. The connection strategy used in this work is to ensure that a collision-free IK solution exists for any point along the edge connecting two nodes. Nodes are connected to other neighbouring vertices within a ball of a specified radius in the workspace. An example graph is visualised in Figure 3.1b.

The poses in Figure 3.1a are generated procedurally by defining a uniform graph of nodes across a volume bounding the bookshelf; however, there exists many possible methods for generating these poses. For example, the manipulator could be teleoperated to various poses or the manipulator could be moved kinesthetically; that is, the human operator could move the end effector directly and store the poses. The requirement is the operator must generate poses such that no islands are formed in the graph. For example, in the bookshelf scenario in Figure 3.1 there is a plane of poses in front of the bookshelf to ensure that there is connectivity between poses within the shelves. However, this could potentially be resolved in a post-processing step automatically, relieving the burden on the operator.



(a) (b)

Figure 3.2: Comparison of first subspaces generated for (a) the 7-DOF fixed-base model and (b) the 6-DOF mobile-base model. Green poses indicate areas of task space that lie within the defined subspace; blue poses indicate the remaining unmapped areas. The 7-DOF is capable of almost full task-space coverage with a single subspace, in contrast to the 6-DOF, due to its kinematic redundancy.

3.5 Motivation for Multiple HAs

If many undefined node mappings remain after running an iteration of the modified Dijkstra procedure, the trajectory library may not cover large regions of the task space. This has an adverse impact on online planning as IK solutions of online tasks in these regions will be far from the trajectory library configurations and thus non-smooth trajectories may be executed as a result.

Node mappings may be left undefined in an iteration of Algorithm 3.3 for several reasons. The environment may have regions that can only be reached without collision by performing large configuration changes over small distances. An illustration of this was depicted in the earlier 2D example in Figure 1.2. Alternatively, the edge cost may be designed to favour subspace mappings that allow the arm to move comfortably without approaching joint and manipulability limits, as was the case for the task-space control applications. This can consequently lead to less coverage of the task space with any single HA.

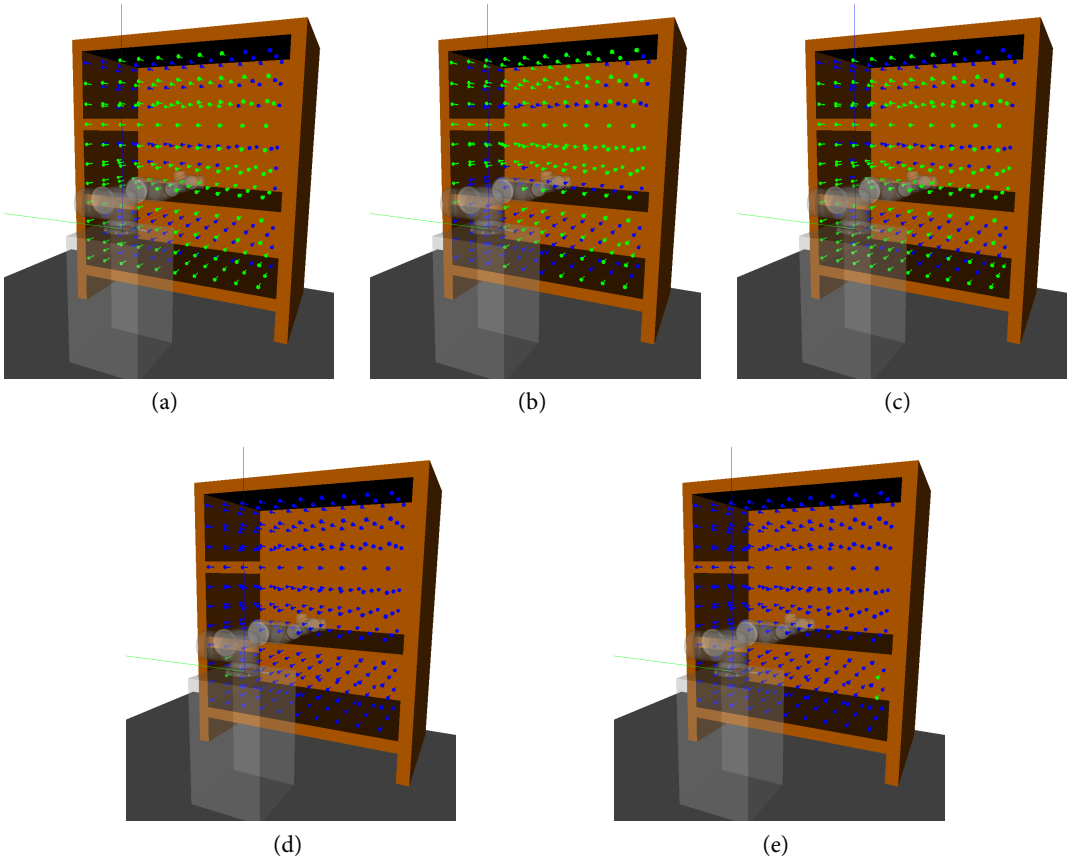


Figure 3.3: Visualisation of task-space subspaces for HAP-Multi. Green and blue poses are defined as in the previous figure. Subspaces are sorted by order in which they were found by Alg. 3.2. Subspaces in (a) and (b) achieve large and diverse coverage, while the subspace in (c) is similar to (a). Subspaces in (d) and (e) cover only small isolated regions of the task space in the left bottom and middle shelf rows, respectively.

Furthermore, the arm's kinematic configuration may contribute to the cause of this. For example, a six-DOF arm has less flexibility in available configurations when moving its end-effector through the workspace in comparison to a seven-DOF arm. This is further compounded in complex environments, as illustrated in Figure 3.2, where the largest HA found for the six-DOF UR5 has less coverage than the seven-DOF sawyer for the same environment and task space.

This therefore motivates the need to search for multiple maps by repeating Algorithm 3.3 for a number of iterations. With each iteration, a new subspace $\hat{T}^i \subset \hat{T}$ and map θ^i to a subspace $\hat{C}^i \subset C$ is found, increasing overall coverage of $\hat{T}$.

The HA allocation for "HAP-multi" is shown in Figure 3.3(a)-(e), where each visualisation

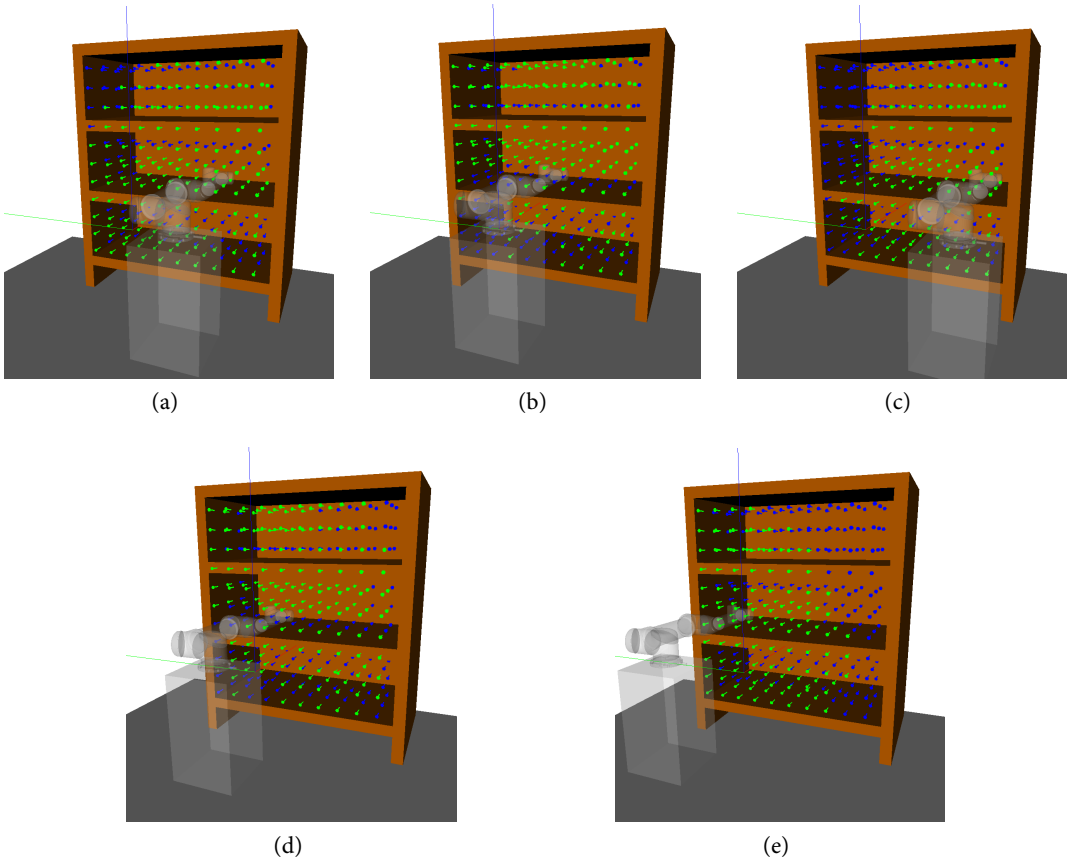


Figure 3.4: Visualisation of task-space subspaces for HAP-Mobile. Note the base pose changes for each subspace. Subspaces are sorted by order in which they were found by Alg. 3.2. Subspaces exhibit large and diverse coverage for all base positions.

shows the HA found in a single iteration of Algorithm 3.3. As can be seen in Figure 3.3a the first HA found is different to the second in Figure 3.3b, particularly around the middle row of the bookshelf where the arm has to perform a large movement of the wrist to reach the poses close to the separating shelves. Figure 3.3c is similar to Figure 3.3a with some additional coverage of the back right of the middle shelf region while sacrificing coverage of poses in the bottom shelf region.

The remaining HAs have only a small number of allocated poses. These small isolated subspaces are particularly difficult for the arm to reach and are unable to be transitioned to from the other regions easily. One can imagine many more of these smaller subspaces existing until coverage of the entire task space. There is some overlap between the HAs in task space which is beneficial as it provides more flexibility during the online planning phase when allocating tasks to HAs.

HAs found for a mobile base case with the same bookshelf environment are visualised in Figure 3.4. Here the mobile base can translate in the horizontal axis parallel to the bookshelf. Each HA has large and diverse coverage which allows for greater flexibility when assigning tasks in the online planning.

Figures 3.4(a)-(e) show the order in which HAs were found in the optimisation routine and as can be seen the base positions that give greater coverage of $\hat{T}$ are considered first. Base positions closer to the middle of the bookshelf have greater coverage while base positions closer to the edge of the bookshelf are able to reach into the back corners of the shelves. The HAs allocated by the algorithm are not easily predictable for implementation of a naive approach such as "Naive-mobile" (see Figure A.1), as evidenced by the complex subspace boundaries in Figure 3.4.

3.6 Analysis

Here a complexity and bounded suboptimality analysis of the HAP algorithm is carried out. For the latter, it is proven that the maps found by HAP are indeed ϵ-GHAs.

3.6.1 Complexity Analysis

For a graph $\mathbb{G}$ with total number of nodes $|\hat{T}|$ and edges $|E|$ the time complexity for Dijkstra with priority queue implementation is $O(|E| + |\hat{T}| \log |\hat{T}|)$ [172]. In *Modified Dijkstra* (Algorithm 3.3) additional overhead time costs are incurred for finding a unique mapping $\theta(t)$ for each task $t \in \hat{T}$, as stated in Lemma 3.1. This is then used in Theorem 3.1 to state the runtime complexity of the algorithm. We then remark on the effect of multiple HAs and utilising a mobile base on the runtime complexity.

Lemma 3.1 (Runtime of *Get Candidate* (Algorithm 3.4))**.** *Runtime complexity of Get Candidate is*

$$O(|\hat{T}||Q(t)|),$$

where $|Q(t)|$ *is the largest set of IK solutions for a single task* $t \in \hat{T}$*.*

Proof. In Algorithm 3.4 all IK solutions are iterated over and those within a specified distance ϵ from the expanded node are kept. Furthermore, edge costs, and optionally task specific costs, must be computed for each one of these solutions. □

Theorem 3.1 (Runtime of *Modified Dijkstra* (Algorithm 3.3))**.** *Runtime complexity of Modified Dijkstra is*

$$O((|\hat{T}||Q(t)| + |E| + |\hat{T}|\log|\hat{T}|)|Q(t)||\hat{T}_{root}|),$$

where $|\hat{T}_{root}|$ *is the number of sampled root nodes.*

Proof. Combing Lemma 3.1 with the time complexity of the original Dijkstra algorithm with priority queue, the runtime of a single iteration of Modified Dijkstra is $O((|\hat{T}||Q(t)| + |E| + |\hat{T}|\log|\hat{T}|)$. This is repeated for each IK solution of a root node and for all root nodes in $\hat{T}_{root}$, resulting in the above runtime. This runtime complexity doesn't include initialisation steps, such as computing IK solutions for all nodes, generating data structures such as k-d trees and undirected graph $\mathbb{G}$, and initialising mapping and path costs. □

Remark 3.1 (Runtime dependence on number of mappings)**.** The runtime of the library construction algorithm simply scales linearly with the number of HAs. In the worst case the number of HAs is equal to $|\hat{T}|$. However, this is unlikely and as described in Section 3.4.3 a more practical approach is to choose a maximum number of subspaces. Furthermore, after the first HA has been found if a ζ is chosen to prune the remaining candidate root node IK solutions then the runtime can be reduced significantly for multiple HAs.

Remark 3.2 (Runtime with mobile base)**.** For the case of assigning HAs to a set of discrete mobile base poses the number of iterations is $\sum_{i=1}^{K} i$. This is due to the fact that the number of candidate base poses is reduced by one after each iteration.

3.6.2 Bounded Suboptimality Analysis

In this section, we show that the trajectories found by Algorithm 3.2 approximate geodesics in the configuration space with bounded error. In other words, the trajectories found have small lengths d_C, and are hence efficient and free of jerky motion. This is because, as we

show, ϵ-GHAs approximately preserve geodesics between two spaces, in our case the task and configuration spaces. We first verify that the map θ found by Algorithm 3.2 is indeed an ϵ-GHA.

Theorem 3.2 (θ is an ϵ-GHA)**.** *θ found by Algorithm 3.3 is an ϵ-GHA.*

Proof. Pick any $t \in \hat{T}$. Then $\forall u_1 \in \mathcal{N}_t$, we have by construction in Algorithm 3.3 the following:

$$|d_C(\theta(t), \theta(u)) - d_T(t, u)| < \epsilon,$$

for some $\epsilon > 0$. Then, $\forall u_2 \in \mathcal{N}_{u_1}$, i.e. the next-nearest neighbours of t, we again have by construction:

$$|d_C(\theta(u_1), \theta(u_2)) - d_T(u_1, u_2)| < \epsilon.$$

Similarly for all $(N-1)$th and Nth nearest neighbours of t,

$$|d_C(\theta(u_{N-1}), \theta(u_N)) - d_T(u_{N-1}, u_N)| < \epsilon.$$

Then, using the triangle inequality we get

$$\begin{aligned}
&|d_C(\theta(t), \theta(u_N)) - d_T(t, u_N)| \leq |d_C(\theta(t), \theta(u_1)) - d_T(t, u_1)| + \\
&\textstyle\sum_{i=1}^{N-1} |d_C(\theta(u_i), \theta(u_{i+1})) - d_T(u_i, u_{i+1})| \\
&< \epsilon + (N-1)\epsilon \\
&= N\epsilon
\end{aligned}$$

As ϵ is arbitrary, $\epsilon = N\epsilon$ gives the required result. □

Furthermore, θ maps shortest paths in the workspace to paths in configuration space that are of bounded length. That is, minimising geodesics are approximately preserved under the mapping. Intuitively, minimising geodesics (referred to henceforth as geodesics) are a generalisation of 'straight lines', or shortest paths, in Euclidean space to more general spaces. The HAP framework constructs θ based on the distance between graph nodes in task and configuration space. To prove that θ approximately preserves geodesics, we are therefore interested in a concept of 'straight lines' on a metric space.

A metric space (X, d_X) is a set X equipped with a metric, or 'distance function', $d_X : X \times X \to \mathbb{R}$ that satisfies the axioms of positiveness, symmetry, and triangular inequality [173]. Drawing upon concepts from metric geometry, we characterise geodesics on metric spaces as follows:

Definition 3.2 (Geodesics)**.** *Given a metric space (X, d_X) with intrinsic metric d_X, a path $\gamma : [0, 1] \to X$ is a geodesic iff:*

$$d_X(\gamma(0), \gamma(1)) = \sum_i d(\gamma(s_i), \gamma(s_{i+1})), \tag{3.10}$$

for any $\{s_i\} \subset [0, 1]$.

Intuitively, the lengths of segments in a geodesic sum up to that of the whole.

Our analysis shows that θ, being an ϵ-GHA, preserves geodesics approximately. Thus, trajectories within a subspace generated by Algorithm 3.2 are of bounded length in configuration space. The practical implication of this is that trajectories executed by HAP are biased to subspaces of the task space that the manipulator can move fluidly within, without requiring any large joint re-configurations.

Theorem 3.3 (ϵ-GHAs preserve geodesics)**.** *Let (X, d_X) and (Y, d_Y) be two metric spaces, and $\theta : X \to Y$ an ϵ-GHA. Then, for any geodesic γ on X, we have:*

$$|d_Y(\theta(\gamma(0)), \theta(\gamma(1))) - \sum_i d_Y(\theta(\gamma(s_i)), \theta(\gamma(s_{i+1})))| \leq (N + 1)\epsilon. \tag{3.11}$$

In other words, the path $\theta \circ \gamma$ is $(N + 1)\epsilon$ away from being a geodesic in the configuration space.

Proof. Applying the ϵ-GHA condition to the end-points of γ, we have:

$$|d_Y(g(\gamma(0)), g(\gamma(1))) - d_X(\gamma(0), \gamma(1))| \leq \epsilon. \tag{3.12}$$

Because γ is a geodesic, we can replace the $d_X(\gamma(0), \gamma(1))$ term in (3.12) with the sum of lengths along any test points $\{s_1 ... s_N\}$:

$$|d_Y(g(\gamma(0)), g(\gamma(1))) - \sum_i d_X(\gamma(s_i), \gamma(s_{i+1}))| \leq \epsilon. \tag{3.13}$$

Using the ϵ-GHA condition on summands individually, we have:

$$\begin{aligned} &\sum_i d_Y(g(\gamma(s_i)), g(\gamma(s_{i+1}))) - (N+1)\epsilon \\ &\leq d_Y(g(\gamma(0)), g(\gamma(1))) \\ &\leq \sum_i d_Y(g(\gamma(s_i)), g(\gamma(s_{i+1}))) + (N+1)\epsilon. \end{aligned} \tag{3.14}$$

$N+1$ arises because there are N epsilons inside the sum and one outside. □

3.7 HAP Evaluation

3.7.1 Experimental Setup

We test the performance of HAP in simulated experiments with kinematically accurate models of a seven-DOF Sawyer [174] and a six-DOF UR5 [175] on a mobile base. Two different application settings are considered. The first is a bookshelf environment, shown in Figure 3.5, where the task of the robot is to simply move the end effector to a set of given 6D poses. The task itself is abstract and could represent, for example, a sequence of viewpoints or grasp poses. The second setting is task space control, illustrated in Figure 3.8, required in applications such as visual servoing. For all experiments, we use Google's or-tools [176] for solving the TSP(s) in (3.4).

In each experiment, we simulate 10 online tasks through random sampling. The tasks are initially sampled from a uniformly random distribution. For fair comparison, the samples are kept only if there exists a collision free IK solution, and, for the mobile base case, if it is reachable by the arm when the base is at the home location which is chosen to be the

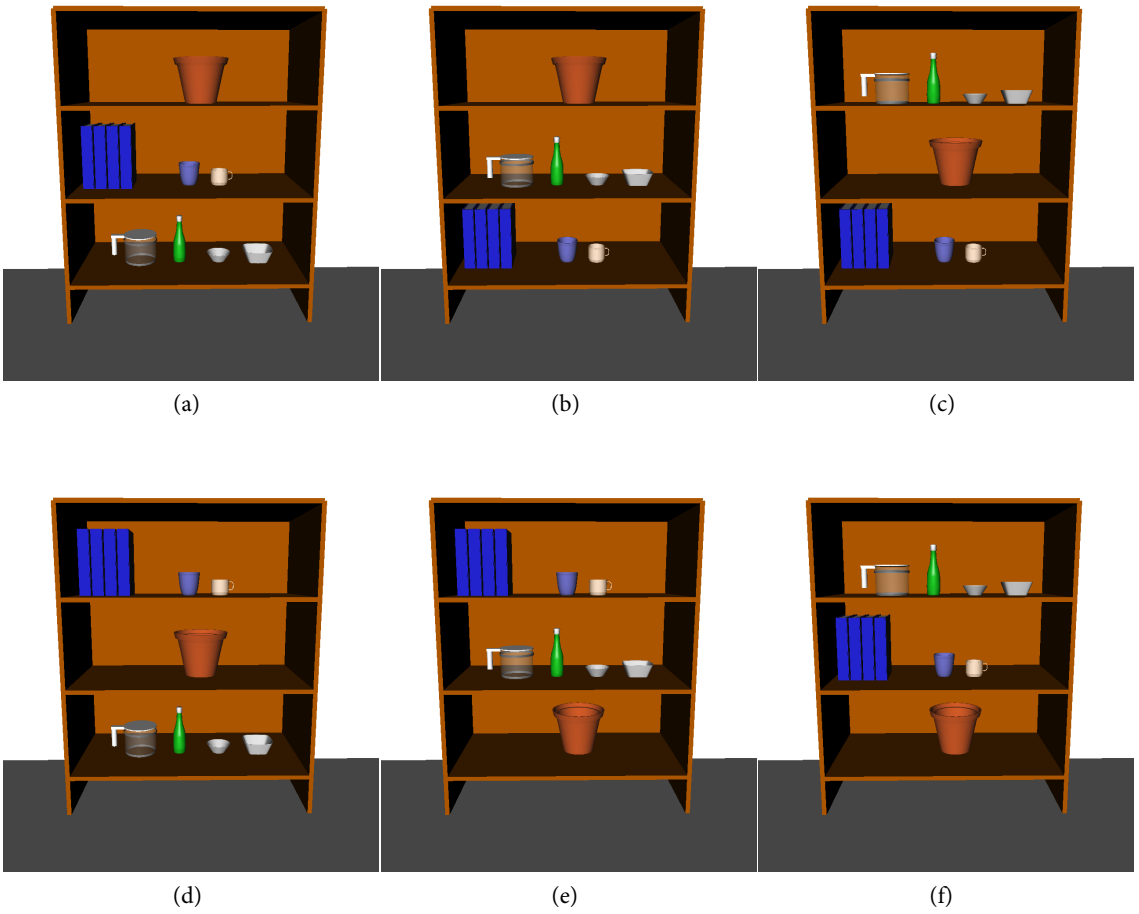


Figure 3.5: Bookshelf object configurations used for the simulation experiments.

centre base pose. Further, we discard samples if there is an existing task within a distance of 0.15m, to ensure realism and wide coverage of the workspace.

We compare the performance of HAP against a naive planner. The naive planner sequences tasks based on their Euclidean workspace distance. Goal configurations for each task are determined greedily at online planning time by choosing the IK solution with the minimum L_2 distance in configuration space to the current robot configuration.

For motion motion planning TrajOpt [177] is initially tried. If TrajOpt fails to find a plan, BIT* [178] is given two seconds to re-plan from scratch while biasing its search towards the same goal configuration as the one used in the TrajOpt attempt. For the naive planner the trajectory optimisation is seeded with ten linearly spaced configurations along a straight line in configuration space.

Variant	Sequencer		Planner	
	HAP	Naive	TrajOpt	BIT*
HAP-Full	✓		✓	✓
HAP-Opt	✓		✓	
Naive-Full		✓	✓	✓
Naive-Opt		✓	✓	
Hybrid-Full	✓	✓	✓	✓

Table 3.1: Variants used for the Sawyer experiments.

Variant	Base		HAs	
	Fixed	Mobile	Single	Multiple
HAP-Mobile		✓		✓
HAP-Single	✓		✓	
HAP-Multi	✓			✓
Naive-Static	✓		—	—
Naive-Mobile		✓	—	—

Table 3.2: Variants used for the UR5 experiments.

For the Sawyer experiments, we find a single map θ, and examine the performance effects of the proposed sequencing and planning methods with the following five variants: "HAP-Full", "HAP-Opt", "Naive-Full", "Naive-Opt", and"Hybrid-Full". These variants differ in the sequencer and planners used and are summarised in Table 3.1. HAP-Full and Naive-Full variants use both TrajOpt and BIT* for motion planning. For the Hybrid-Full variant a mixture of HAP and the naive planner is employed. Here the naive workspace distance is used for sequencing, while the post-processing optimiser uses the library trajectories as the initial condition. All variants use the same home configuration based on the average configuration of the single HA.

For the mobile UR5 experiments, we demonstrate the performance benefit of finding multiple maps $\{\theta_i\}_{i=0}^{K}$ with the following five variants: "HAP-Mobile", "HAP-Multi", "HAP-Single", "Naive-Static" and "Naive-Mobile". These variants all use the same planner configuration as the HAP-Full variant in the Sawyer experiments. As before, prefixes here indicate the sequencer used for each variant (HAP or naive). The variants additionally differ in the type of base (fixed or mobile) and, for HAP variants, whether a single or multiple HAs are used for trajectory library generation.

Each variant's properties are summarised in Table 3.2. For the Naive-Mobile variant, the

base is mobile and we pre-allocate task poses to three different base positions, as illustrated in Figures A.1 and A.2. If an IK solution cannot be found for a task pose with the base position away from home, the task pose is reallocated to the home base position. The home configuration used for the naive variants is the same as "HAP-mobile".

The performance of each variant is averaged over 50 scenarios with different randomly generated tasks. The following metrics are evaluated:

- *Planning success rate* refers to the percentage of tasks for which the motion planner succeeded in finding a collision free path. In the case that BIT* is used to fall back on, a failure occurs if a valid plan cannot be found within the two second time limit.
- *Average planning time* refers to the average computation time taken for the motion planners to compute a plan for a task.
- *Average execution time* refers to the average time taken for a successful execution, assuming the arms are operating at their maximum joint velocities.
- *Sequencing time* refers to the time taken to match library trajectories, adapt them and then compute a sequence, including the IK solution computation for each task for both the naive and HAP planning methods.

All metrics above include the trajectories for switching between subspaces (i.e. to and from the home configuration).

Open Robotics Automation Virtual Environment (OpenRAVE) [179, 180] is used for the simulation environment with several third party plugins: motion planning library bindings [181–183], unified robot description format (URDF) interface [184], and flexible collision library (FCL) interface [185, 186]. The UR5 and Sawyer robot arm models and control libraries are from the official Github repositories [187, 188].

3.7.2 Implementation Details

The parameter choices are summarized in Table 3.3. All nodes are initialised with path cost c_{max} which is set to 5.0. For all multiple HA experiments the optimisation routine

d_C	L_∞
d_T	L_2
ϵ	0.35 (Sawyer)
	0.85 (UR5)
$\lvert\hat{T}_{root}\rvert$	10
c_{max}	5.0
ρ	2.0
ρ_s	0.02
k	10

Table 3.3: Parameters used for experiments.

is terminated once a maximum of five subspaces are found. The subspace exploration penalty ρ is chosen to be 2.0. This is chosen in combination with the c_{max} such that by the fourth iteration, if a node has been mapped in all previous iterations then it will be set to undefined regardless of its true edge cost. For the mobile base experiments the base positions are evenly spaced between $[-0.4, 0.4]$m along the y-axis, which is parallel to the bookshelf.

For the online planner, the number of closest neighbours used when matching library tasks is 10. The threshold used for terminating the search over mappings when matching a library configuration is an L_2 distance of 0.7 and 0.5 radians for the bookshelf and task-space control experiments, respectively.

The penalty ρ_s for biasing subspaces to be near each other is set as 0.02. The home pose used for the Sawyer experiments is shown in Figure 3.1c and is the same for the UR5 experiments.

3.7.3 Bookshelf Environment with 6D Pose Tasks

To benchmark HAP, extensive comparison experiments are first carried out using the bookshelf configuration in Figure 3.5a. Then to validate that there is no overfitting to one particular object configuration, the experiments are repeated on the other bookshelf configurations in Figure 3.5 using only the best performing HAP variants. The computed HAs are shown and discussed later in Section 3.5.

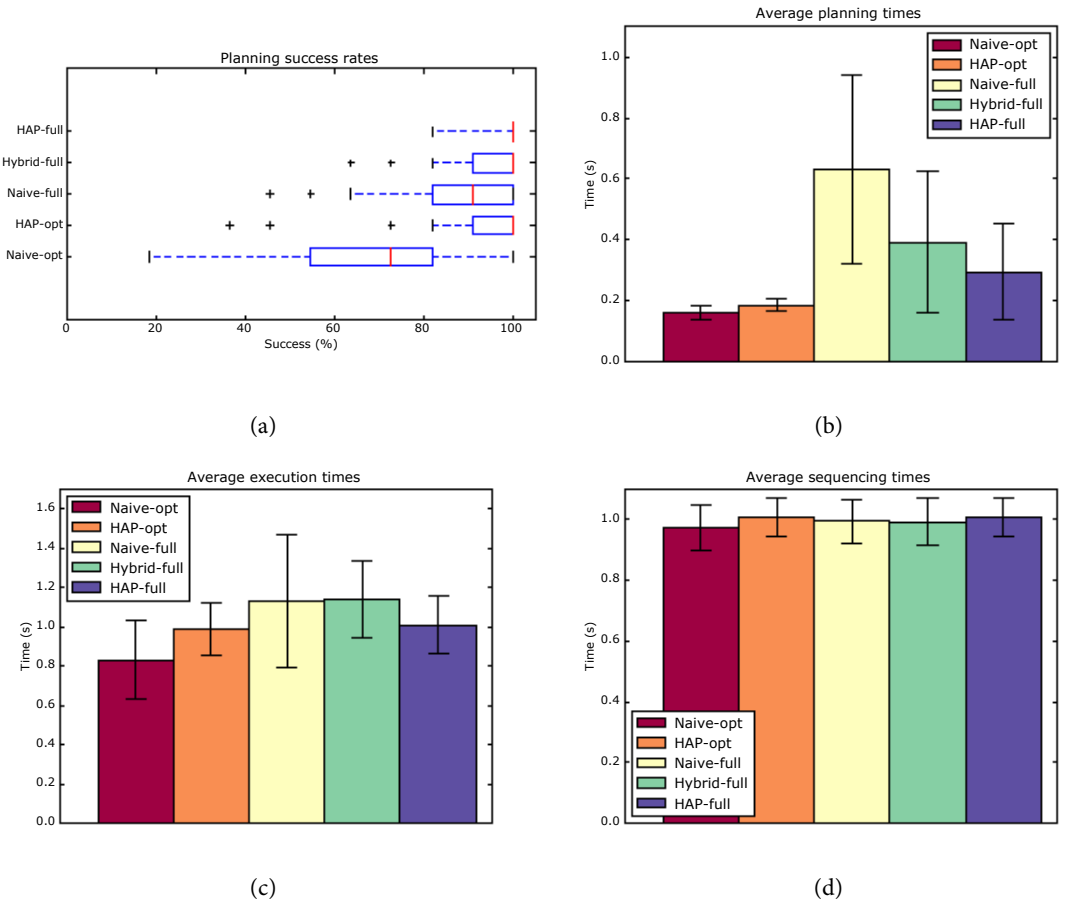


Figure 3.6: Planning success rate and times for 7-DOF Sawyer experiments with the fixed-base model. Mean and standard deviation for sequencing, planning, and execution time refers to a batch size of 10. HAP variants achieved higher planning success rate and faster plan times compared to the baseline naive variants while achieving similar or better execution times.

3.7.3.1 Sawyer

Here we present results from the experiments performed on the bookshelf environment in Figure 3.5a with a simulated Sawyer arm. For the Sawyer it was only necessary to utilise one HA to achieve good performance. The results for the Sawyer bookshelf experiments are shown in Figure 3.6.

HAP-Full heavily outperforms both naive planner variants in terms of planning success rate while achieving a similar success rate to other HAP variants yet with lower variance. The HAP-Opt variant achieves lower planning and execution times than HAP-Full with

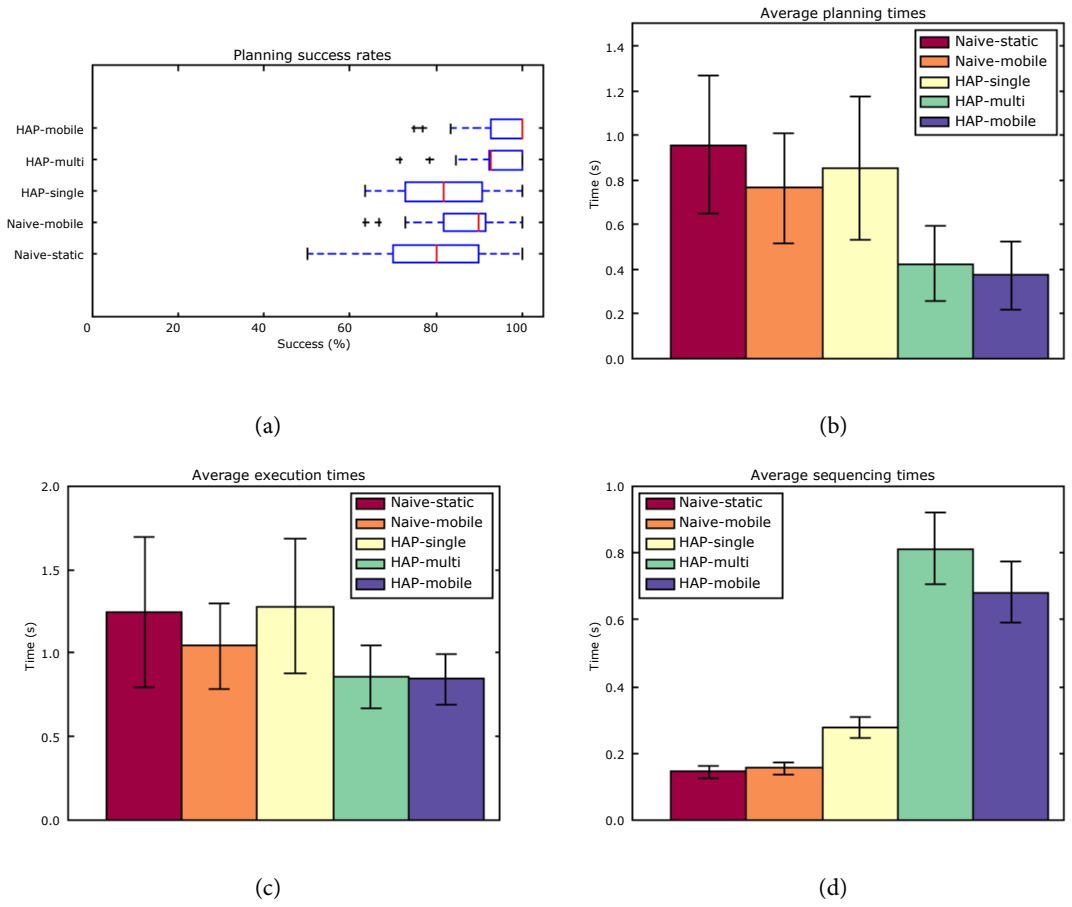


Figure 3.7: Planning success rate and times for 7-DOF UR5 with the mobile-base model. Results are consistent with fixed-base experiments shown in Figure 3.6.

slightly worse planning success, due to the absence of BIT* to fall back on, making it a viable choice if these metrics are more important for the given application. Naive-Full and Naive-Opt variant perform poorly with very low success rates and, for the former, high planning times. The Hybrid-Full variant performs worse in all measured metrics compared to the other HAP variants, suggesting that the trajectory library costs play an important role in planning and task sequencing.

For task sequencing, the naive planner on average took 0.97 seconds to compute whereas HAP took 1.01 seconds, including trajectory library search and adaption. This overhead is negligible when considering the significant increase in performance gained. In fact, for both planners the majority of computation time is from generating the IK solutions for each task.

3.7.3.2 UR5 with mobile base

Similar experiments were carried out for a UR5, again on the environment in Figure 3.5a. The results for the experiments are shown in Figure 3.7. For these experiments the UR5 is on a mobile base which can translate in the y-axis which is parallel to the bookshelf. Here, multiple HAs are utilised and results are compared with utilising only the first HA to motivate their necessity.

Although HAP-Mobile has slightly better performance than HAP-Multi – in particular, it achieved a median planning success of 100% versus 93% – it is expected that they would have similar performance in this setup since all tasks are reachable from the centre base position. Both methods had a similar average number of subspace switches with 2.74 versus 2.88, for HAP-Mobile and HAP-Multi respectively. This suggests that both methods are utilising most of the subspaces found. However, another benefit which is not quantified here is that having a mobile base is more flexible in that the arm has greater reachability and can be extended to larger workspaces.

As expected, Naive-Static and Naive-Mobile variants performed worse in all measured metrics except for sequencing time. The average number of subspace switches for the latter was 1.66 which is lower than HAP-Mobile and HAP-Multi. However, given the poor planning success this suggests that there is a need for more complex subspace allocations, such as those found by HAP. HAP-Single performed similarly to the naive methods suggesting that multiple subspaces are necessary for any sort of performance gain for a non-redundant arm such as the UR5.

The sequencing overhead for HAP with multiple subspaces is apparent in Figure 3.7d; however, averaged over the ten tasks this is negligible in comparison to the time savings in planning and execution times. If necessary, the sequencing time could be easily reduced further by parallelising the IK computation for each subspace and even for each individual task.

Environment	a	b	c	d	e	f
Sawyer						
Success rate (%)	97.09 ± 5.59	97.09 ± 5.87	97.09 ± 5.59	95.27 ± 6.36	94.55 ± 9.09	95.64 ± 6.36
Plan times (s)	0.29 ± 0.16	0.30 ± 0.15	0.33 ± 0.16	0.37 ± 0.20	0.42 ± 0.21	0.38 ± 0.20
Execution times (s)	1.01 ± 0.15	0.98 ± 0.15	1.02 ± 0.20	1.02 ± 0.17	1.05 ± 0.19	1.07 ± 0.23
Sequence times (s)	1.01 ± 0.06	1.01 ± 0.06	1.01 ± 0.08	1.01 ± 0.09	1.06 ± 0.08	1.02 ± 0.08
UR5						
Success rate (%)	95.20 ± 6.68	94.57 ± 5.55	93.46 ± 6.56	93.86 ± 5.98	93.04 ± 5.40	94.76 ± 4.96
Plan times (s)	0.37 ± 0.16	0.42 ± 0.18	0.43 ± 0.19	0.43 ± 0.18	0.43 ± 0.19	0.37 ± 0.13
Execution times (s)	0.84 ± 0.15	0.87 ± 0.19	0.85 ± 0.19	0.88 ± 0.15	0.86 ± 0.20	0.81 ± 0.13
Sequence times (s)	0.68 ± 0.09	0.70 ± 0.09	0.70 ± 0.10	0.72 ± 0.08	0.71 ± 0.09	0.69 ± 0.08

Table 3.4: HAP validation results in six instances of the bookshelf environments where object locations are randomly perturbed. Performance is consistent across all environment instances.

3.7.3.3 Validating generalisability

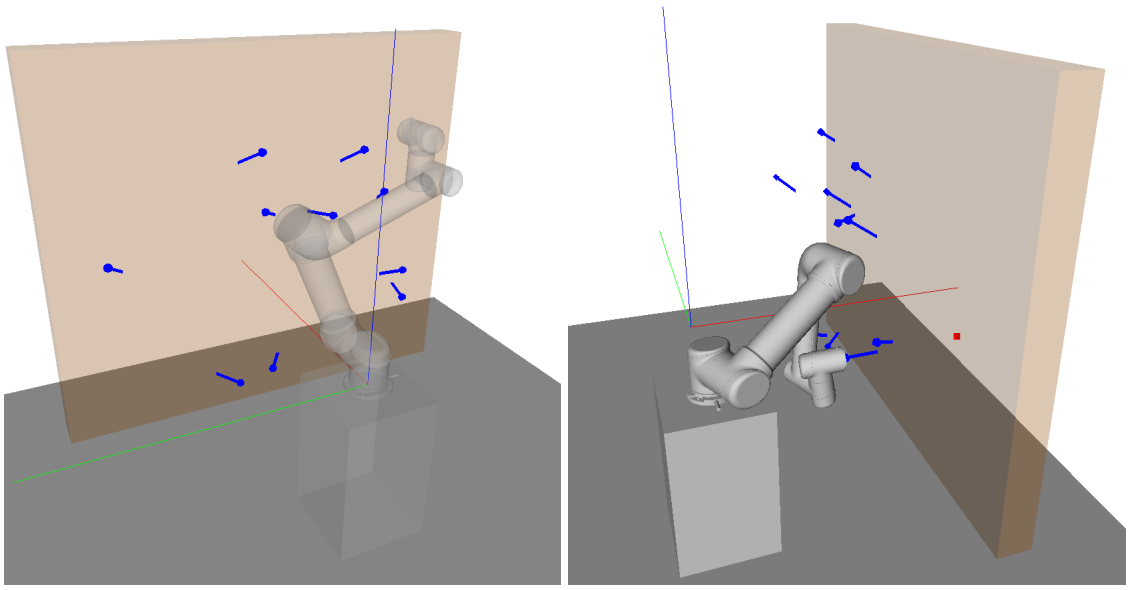
Here we re-run the experiments on the bookshelf configurations in Figure 3.5(b)-(f) using the HAP-Full and HAP-Mobile variants with the Sawyer and UR5 respectively. We average the results over 50 scenarios with ten tasks per scenario, per bookshelf configuration.

The results are shown in Table 3.4 averaged over all the bookshelf environments with the results from Figures 3.6 and 3.7 included under "Environment a" for ease of comparison. As can be seen, the results for both the Sawyer and UR5 match closely with those observed in the previous experiments, validating the generalisability of our method.

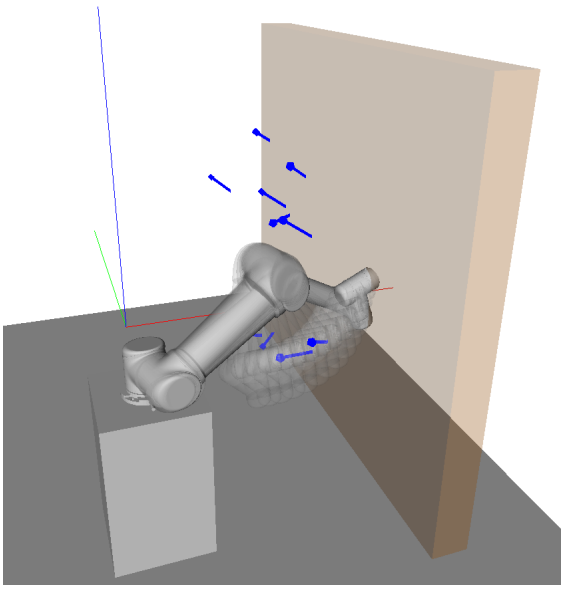
3.7.4 Task-Space Control

To demonstrate the flexibility of HAP the framework was tested on a position-based visual servoing (PBVS) application [189]. The experimental setup is shown in Figure 3.8. This experiment is designed to demonstrate that even with a simple environment HAP can still be useful when the tasks themselves are complex. The task here is to plan the arm to an approach pose, which is a fixed offset from the target, and then use task-space control to move the end-effector in a linear motion towards the target.

This scenario is typical of many PBVS scenarios where the object to be grasped or inspected is within clutter which is computationally expensive to model and plan through. Hence, planning to the approach pose considers an simplified bounded volume obstacle;



(a) (b)



(c)

Figure 3.8: Task-space control experiments: (a) shows the experimental setup and an instance of tasks, (b) shows the arm at the approach pose (blue vector) facing the initial estimate of target position (red dot), (c) shows the trajectory produced by the PBVS controller.

however, collisions with the bounded volume are ignored when servoing towards the target. Such an approach was taken in a pruning robot application [17]. Another reason

for approaching from an offset may be to perform corrective measurements of the target's location.

Once the arm has reached the approach pose, task space control [78] moves the arm's end-effector linearly towards the target. Let $e_m \in \mathbb{R}^3$ be the current position of the end effector and $e_t \in \mathbb{R}^3$ the position of the target. Let α_m be the current orientation of the end effector and α_t be the desired target orientation in vector form. The latter is a unit vector in the global frame that we want the end-effector's local "forward-facing" axis to align with. The desired task-space velocities to move the end effector towards the target is given by:

$$\dot{e_d} = k_e(e_t - e_m) \tag{3.15}$$

$$\dot{\alpha_d} = -k_\alpha(\alpha_t \times \alpha_m)\cos^{-1}(\alpha_t \cdot \alpha_m) \tag{3.16}$$

where $k_e, k_\alpha \in \mathbb{R}$ are positive gains, $\cdot$ is the dot product and $\times$ is the cross product. The desired joint velocities to move the end effector towards the target is then given by:

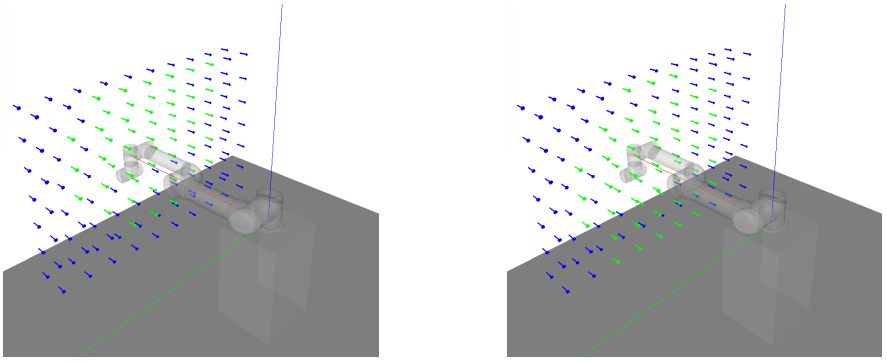
$$\dot{q_d} = \mathbf{J}^+ \begin{bmatrix} \dot{e_d} \\ \dot{\alpha_d} \end{bmatrix}, \ \mathbf{J}^+ = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^{-1}. \tag{3.17}$$

$\mathbf{J}^+$ is the pseudoinverse of the Jacobian $\mathbf{J}$ and is well-defined even when the Jacobian is rank-deficient. During this process, it is essential to ensure that the robot avoids its joint limits and any singularities. We thus use the following task-specific cost as per Section 3.4.3.2 that consists of two weighted terms:

$$\begin{aligned} c(\theta(v), \theta(u)) = \xi_{lim} \frac{1}{K} \sum_{j=1}^{K} \tanh\left(2\left|\frac{\theta(u)_j - q_j^{lower}}{q_j^{upper} - q_j^{lower}} - \frac{1}{2}\right|\right) \\ + \xi_{manip}\kappa(J), \end{aligned} \tag{3.18}$$

where $\kappa(J)$ is the condition number of the Jacobian.

The first term penalises configurations near the joint limits where j is an index to each joint in the manipulator and q_j^{lower} and q_j^{upper} are the lower and upper limits of the j-th manipulator joint, respectively. The second term encourages configurations with higher



(a) (b)

Figure 3.9: Subspaces computed for HAP-Multi in the task-space control experiments. The subspace in (a) provides coverage of the back plane of poses while (b) covers the front plane.

manipulability [190]. We penalise the condition number $\kappa(J)$ because singularity occurs when the condition number $\kappa(J)$ approaches ∞ [191]. Weights ξ_{lim} and ξ_{manip} are chosen to be 0.5 and 0.1 respectively.

We detect and record two failure conditions. The first is what we refer to as a joint limit failure and this occurs when the arm reaches within 0.01 radians of any of the joints' limits. The second is a manipulability failure which occurs when $1/\kappa(J) < 0.015$, indicating that the arm is about to move into a singularity. In this case, we terminate the approach.

For simulation purposes, we assume the target 6D pose is given and add Gaussian noise to the target's position and orientation which decreases linearly as the end-effector approaches the target, as illustrated in Figure 3.8c. When tasks are generated, both the target pose and approach pose are ensured to have at least one valid IK solution.

An approach distance of 0.35m from the target was used. This is an ideal approach distance for visual servoing applications as RGB-D sensors typically have a minimum range of around $0.2-0.3$m and this facilitates making corrective measurements before and during approach. Task space control over such a long approach is challenging and requires the UR5 arm to extend almost to its full reach.

The $\hat{T}$ used for the trajectory library generation and HA allocations for the experiments are shown in Figures 3.9 and 3.10. $\hat{T}$ is chosen such that it covers the region where targets are expected to be located, rather than approach pose locations. Further, as can be seen

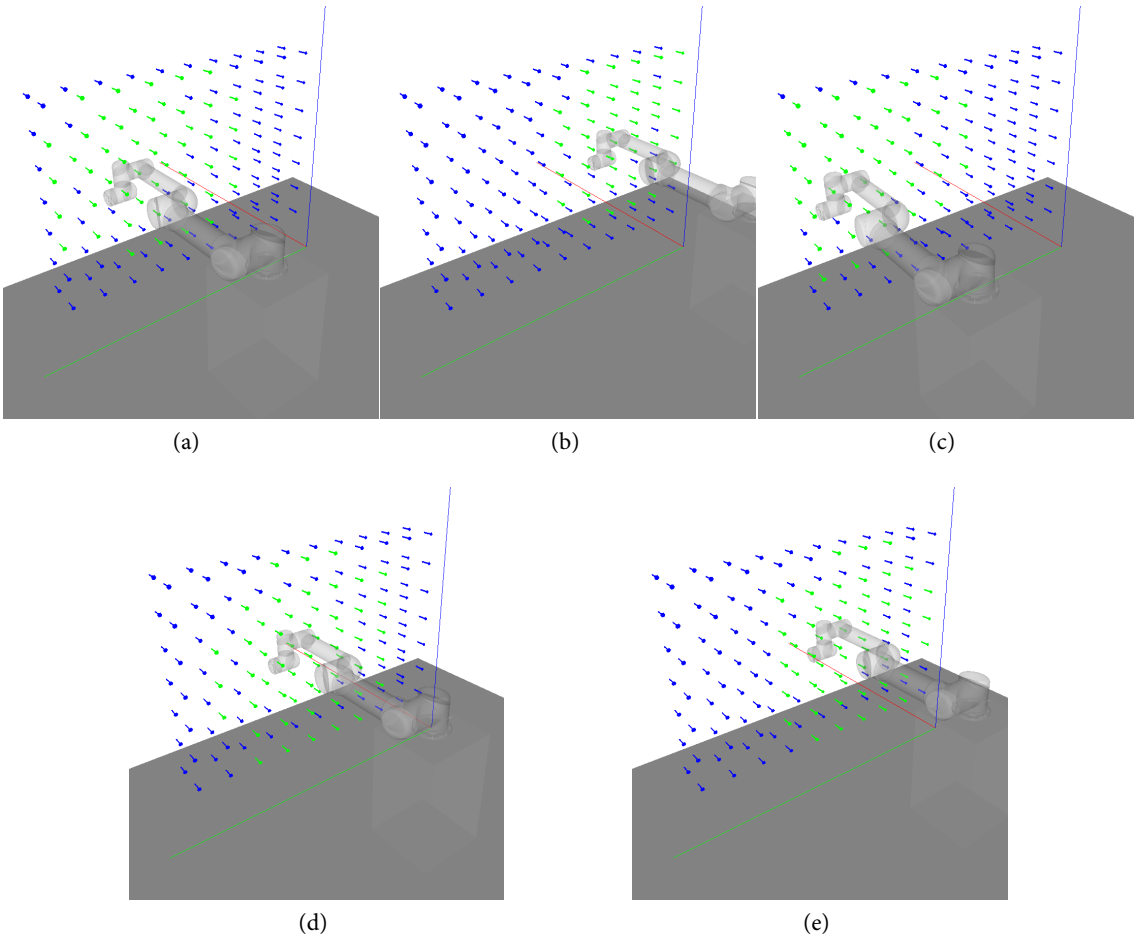


Figure 3.10: Subspaces computed for HAP-Mobile. The five subspaces generated have unique base positions and collectively achieve near-complete coverage of the task space.

the box collision where targets are located is removed. This is because the arm will be ignoring collisions when servoing towards the targets.

Notice that the individual HAs in Figures 3.9 and 3.10 appear to have smaller coverage than expected. This is due to the fact that the modified Dijkstra routine (Algorithm 3.3) might never assign a mapping if the computed edge cost for a task always exceeds c_{max}. In the context of this application this means that some tasks could not be assigned without being undesirably close to joint limits or having low manipulability.

Here, trajectory library matching is slightly modified. First library trajectories are matched using target poses. However, because the arm must plan to the approach poses, the matched library trajectories are adapted to them instead.

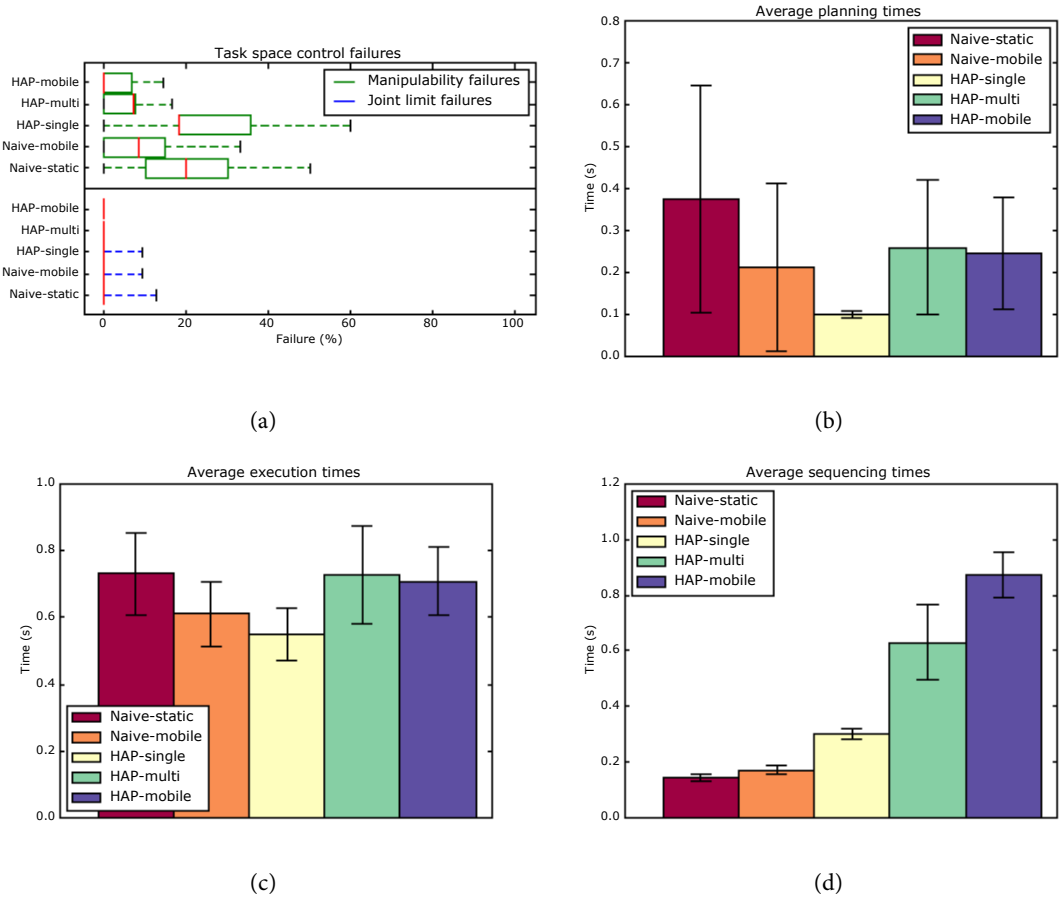


Figure 3.11: UR5 task-space control experiment results.

The results of the experiments are plotted in Figure 3.11. The naive planners and the HAP-Single variant suffer from high task space control failures. HAP-Multi and HAP-Mobile achieve very low failure rates, with slightly lower manipulability failures for the latter. This suggests that having a mobile base is beneficial due to the greater flexibility it provides in how the arm can execute tasks. Impressively, HAP-Multi is able to achieve similar performance to the mobile variant with a static base.

All HAP and naive variants have similar execution times and mostly similar planning times. Notably, HAP-Single has excellent planning and execution times yet high manipulability failures. This suggests that utilising multiple HAs is important for consistently choosing approach configurations that maintain high manipulability. As a result, planning time and trajectory efficiency are traded off for higher control success rate.

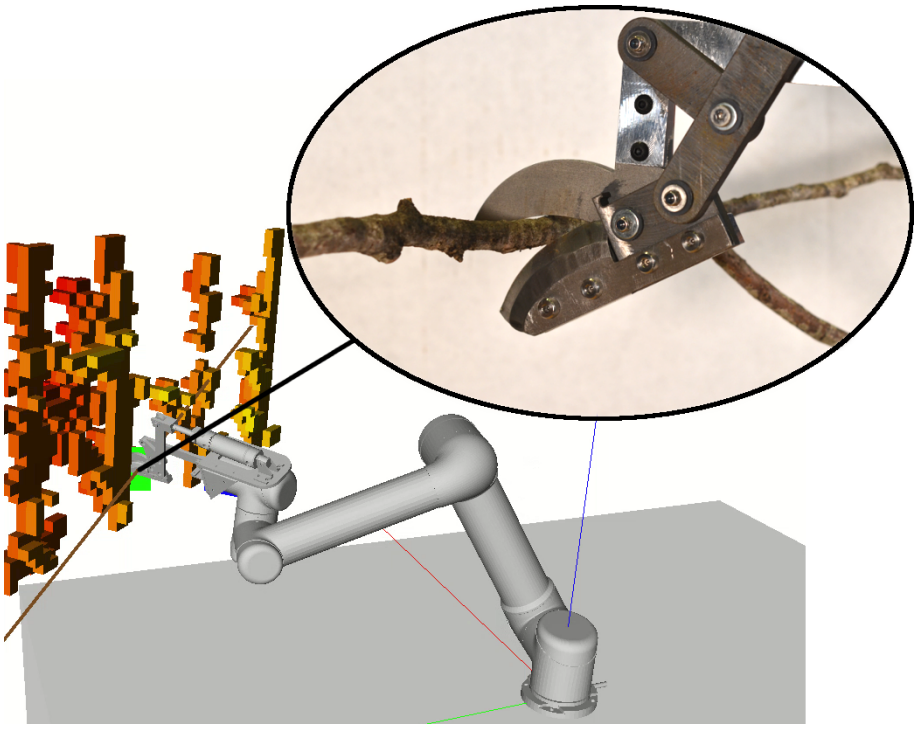


Figure 3.12: Our autonomous pruning system with *(inset)* a custom cutter mechanism.

As in the bookshelf experiments, it is again apparent that having multiple subspaces comes at the cost of higher sequencing times here. As for planning success rate, both HAP and the naive frameworks achieved a median of 100% which is expected given that the environment is simplistic and the focus here is more on the complexity of the control task.

3.8 Robotic Pruning

To demonstrate the flexiblilty of HAP, robotic pruning will be used as a motivating real world example. Due to the precise nature of pruning, that is the arm needs to position its cutter very accurately otherwise it will miss the branch (see Figure 3.12), visual servoing is necessary to deal with sensor and positioning errors.

The implementation details for HAP will be focused on here. For details on the perception subsystem, such as environment representation, cut point generation and processing of sensor data, see [17].

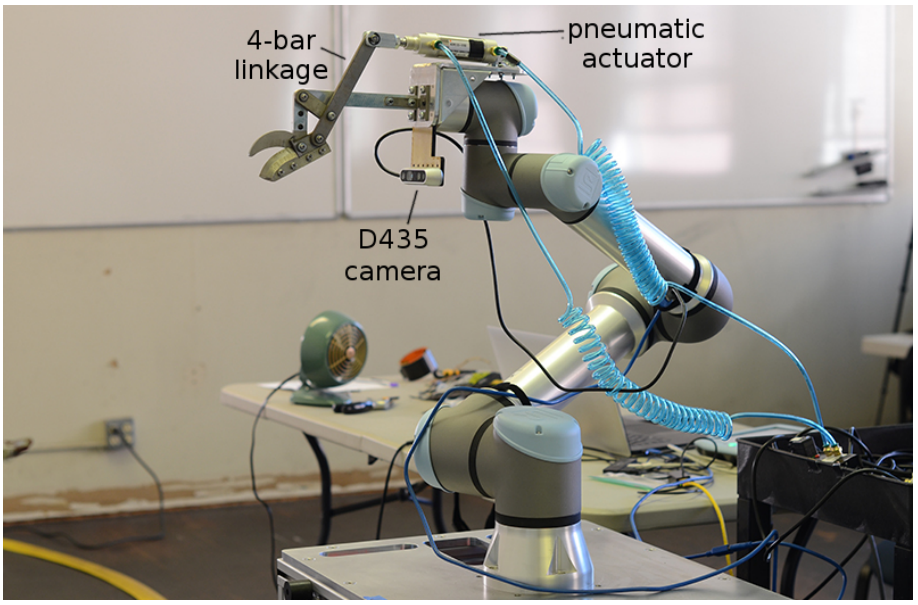


Figure 3.13: Our pruning robot setup, consisting of a UR5e arm, an Intel RealSense D435 RGB-D camera, and custom pneumatically-actuated shears.

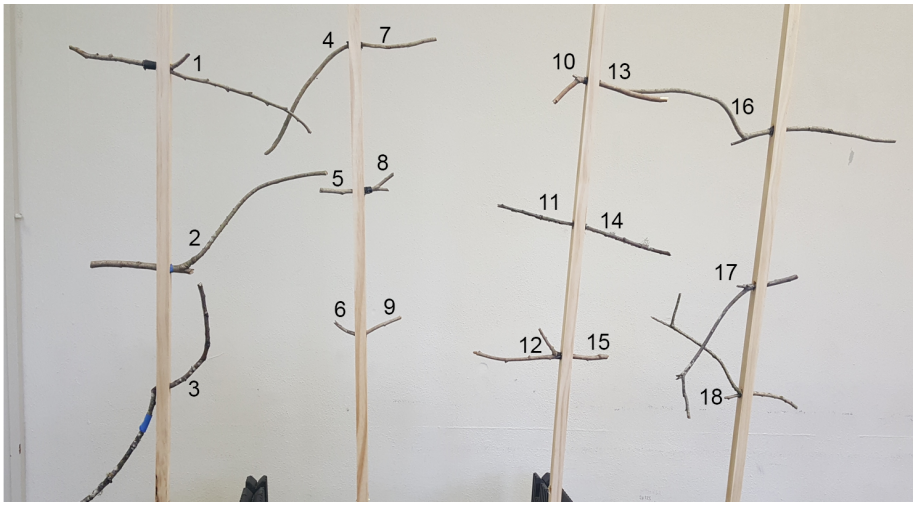


Figure 3.14: The test setup used for experimental evaluation. Possible cut points are enumerated.

3.8.1 Experimental Setup

Our experiments use the hardware setup shown Figure 3.13 which consists of a UR5 robot arm with high-speed pneumatic shears and a RGB-D camera assembly attached to its end effector. The test bed for the experiments is shown in Figure 3.14 which is designed to resemble a UFO sweet cherry training system with horizontal branches emerging from vertical leaders on the tree. For this orchard system, one desirable pruning strategy is

simply to cut the branches extending from the leaders. Similarly, our setup consists of four vertical leaders, each of which has holes at three different vertical offsets to hold the branches used for testing.

We evaluate the performance of an end-to-end pruning system in terms of accuracy, speed and reliability. Similar to before, for comparison we use a "naive planner" that sequences cut points based on Euclidean distance in the workspace and plans trajectories as our method does; however, with no prior trajectory information. Goal configurations for each approach pose are determined greedily at plan time by choosing the closest IK solution to the current robot configuration.

Each experiment involves 6-10 randomly selected cut points chosen from the 18 possible locations shown earlier in Figure 3.14. We randomly configured the branches between experiments to obtain a variety of poses for the planners.

To position the cutter accurately, the system initially moves the cutter to an intermediate *approach pose* and then follows a linear *approach path*. Motion planning is implemented using HAP and we utilize the trajectory costs from the trajectory library to optimize the cut sequence.

There are three stages during which failures can occur: planning to the approach pose, executing the approach path (e.g., if the robot encounters a servoing error), and cutting the branch (e.g., if the mouth of the cutter does not align properly with the branch). An *execution success* is when no failures occur during the first and second stages, whereas a *cut success* is when no failures occur at any stage. Sequencing, planning, and trajectory times are total times for each experiment normalised by the number of execution successes. Approach path failures only consider executions that were able to plan to the approach pose in the first stage.

3.8.1.1 Approach Pose Selection

Our goal in picking an approach pose is to move the cutter to a pose such that the cutter is pointed towards the branch and the branch is roughly orthogonal to the plane containing the cutter. To generate the candidate approach poses, we assume the pitch of the approach

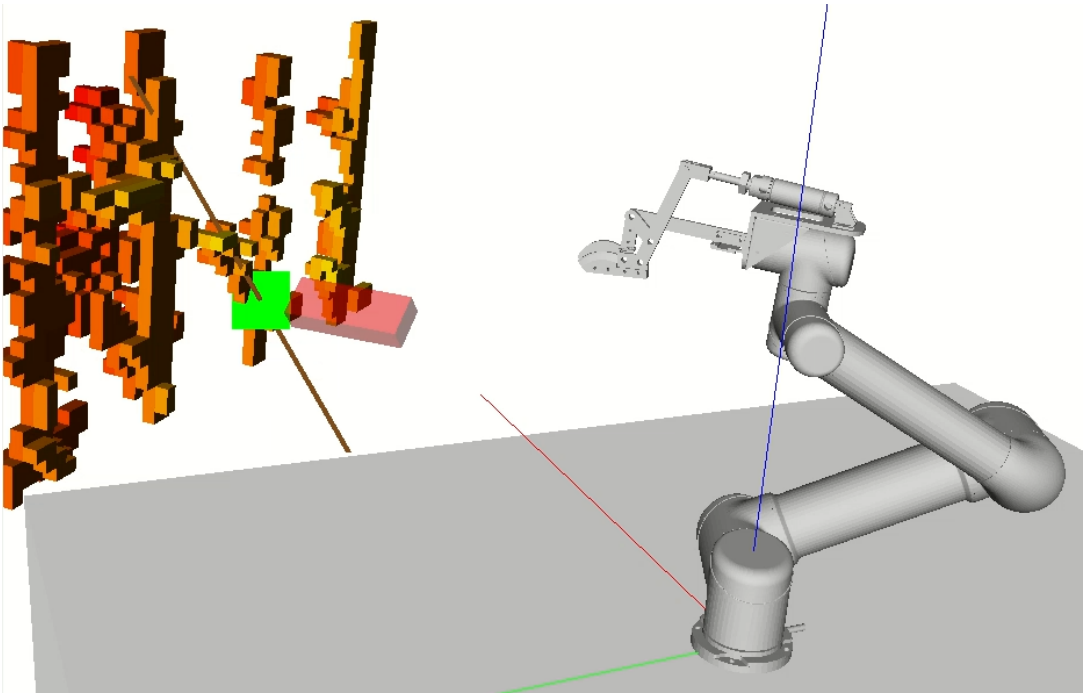


Figure 3.15: An image from the simulation showing the OctoMap representation of the tree system. The target branch is shown as a brown line, and the red box represents the potential collision area with the cutter and the environment on the approach.

pose is zero. We then choose the roll and yaw of the approach pose such that the cutter plane is orthogonal to the branch. If such an approach is not possible, either due to collisions or lack of IK solutions, we vary the yaw within some threshold, chosen to be ± 45 degrees, until a viable approach pose is found or the goal is deemed unreachable and discarded.

To check for collisions along the approach, we create a bounding box as shown in Figure 3.15 which represents the likely path of the cutter during the approach, slightly inflated to account for deviations in the path. We then check if the box collides with W_{env}. The assumption we make is that because the arm is moving on a short linear path, the arm is unlikely to perform any large movements which will collide with the environment or itself, and as such we only need to check for environment-cutter collisions.

		1	2	3	4	5	6	7	8	9	10	Avg
Success rate	Ours	83%	100%	100%	**83%**	100%	**89%**	**88%**	**100%**	75%	100%	**92%**
	Naive	83%	100%	100%	33%	100%	33%	50%	86%	75%	100%	76%
Sequencing time (s)	Ours	0.33	0.23	0.24	0.30	0.22	0.24	0.27	0.21	0.25	0.23	0.25
	Naive	**0.00**	**0.00**	**0.00**	**0.01**	**0.00**	**0.01**	**0.01**	**0.01**	**0.00**	**0.00**	**0.00**
Planning time (s)	Ours	**0.50**	**0.23**	**0.29**	**0.40**	**0.28**	**0.29**	**0.38**	**0.38**	**0.34**	**0.31**	**0.34**
	Naive	0.57	0.41	0.48	1.81	0.46	3.54	2.84	0.47	0.68	0.47	1.17
Trajectory time (s)	Ours	7.63	4.10	3.89	**5.73**	4.29	**5.52**	**5.46**	5.68	**4.40**	4.48	**5.12**
	Naive	**3.34**	**2.88**	**2.93**	16.12	**2.46**	23.95	15.74	**2.87**	4.51	**3.02**	7.78
Throughput (s)	Ours	8.46	4.57	4.42	**6.43**	4.79	**6.05**	**6.11**	6.26	**5.00**	5.01	**5.71**
	Naive	**3.91**	**3.29**	**3.41**	17.94	**2.92**	27.49	18.59	**3.35**	5.20	**3.49**	8.96

Table 3.5: Results from ten experiments. All times are normalised by number of execution successes within each experiment. Our method consistently performs better in terms of success rate and planning times and on average achieved faster throughput.

3.8.1.2 HAP implementation details

For the pruning application, our discretisation strategy is to assume that the cut points lie in some vertical 2D plane and generate a set of uniform 6D poses normal to this plane, similar to Figure 3.9. The edge costs are the same as those used in the task-space control experiments for the HAP evaluation in Section 3.7.4.

3.8.1.3 Approaching the Cut Point

Once the cutter has reached the approach pose, the approach routine moves the cutter linearly towards the goal. We compute the desired linear and angular velocities $\dot{e_d}$ and $\dot{\alpha_d}$, which are now with respect to the centre and orientation of the cutter, and then compute the desired joint velocities using (3.17).

During this process, we use the eye-in-hand camera to update the cut point location. Additionally, we detect if the configuration approaches a joint limit or falls below a given manipulability threshold, in which case we terminate the approach. If the robot does not encounter a servoing failure, it executes the cut once at the goal. Once complete, we reverse the approach trajectory to return the cutter to the approach pose.

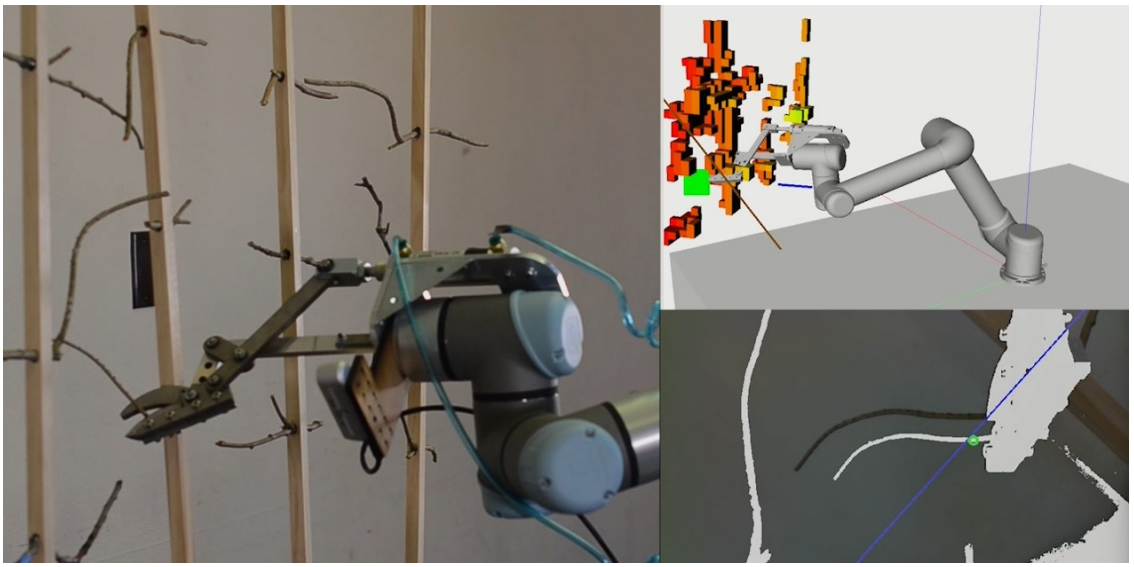


Figure 3.16: Robot pruning a branch with simulation environment (top right) and in-hand camera view (bottom right).

3.8.2 Hardware Results

An example prune is shown in Figure 3.16. As can be seen, the system can handle complex branch orientations and achieve highly precise cutter placement. The environment reconstruction is represented using OctoMap [149], a 3D occupancy grid, constructed from point cloud data from the camera

Table 3.5 shows results averaged within each of the ten experiments as well as total throughput. Our system executed more cuts than the naive planner (92% versus 76% success). Although sequencing times were higher, planning and trajectory times were significantly lower, and approach path failures were lower (1.7% versus 20.2% failures). Independent of planning, the average time of a successful servoing phase, including cutting and rewinding the approach, was about 13 seconds, and the overall cut success rate was 75%.

The results indicate that the system is computationally efficient enough to carry out pruning in real time. They additionally indicate that the system is able to prune with high accuracy while avoiding damaging collisions with the environment. Our system achieved a 1.5x speedup in planning throughput with fewer approach path failures than the naive method.

3.9 Summary

This chapter presented a new motion planning framework for robotic manipulators that is designed to perform efficiently in practice given a user-defined task space. The framework automatically constructs a library of trajectories that move the manipulator's end effector from one pose to another within this task space, and uses this library to quickly produce motion plans online. The library is constructed by finding a set of subspaces with associated ϵ-Gromov-Hausdorff approximations that guarantee short trajectories of bounded length which also can be concatenated smoothly. A distinctive feature of this method is its usefulness in batch-query scenarios where an unordered set of goal poses is provided and the algorithm is free to choose a favourable sequence of minimum total cost. Because all precomputation is performed automatically, this work helps to enable rapid deployment of manipulators for various applications based on the task-space definition.

The proposed framework was evaluated in several experimental scenarios with 6-DOF and 7-DOF manipulators on fixed and mobile bases, including point-to-point motions and long sequences in a bookshelf environment. Results showed notable performance improvement over baseline methods in planning time, planning success rate, and smoothness measured by jerk. Experiments with manipulators mounted to a mobile base demonstrate the generality of the method for a variety of hardware configurations. In the next chapter we utilise HAP within an active perception framework to enable fast and kinematically accurate planning that considers the true motion of the system.

Chapter 4

Targeted Information Gathering

This chapter presents a non-myopic active perception framework suitable for systems with complex motion and sensor models. This consists of a novel targeted information gathering algorithm for balancing between exploration and targeted observations of objects of interest in the scene. This algorithm is used for viewpoint evaluation and integrated with HAP into an MCTS algorithm for computing an optimal sequence of viewpoints. The framework is evaluated in a real hardware experiment where a manipulator is tasked to find and then grasp clustered and occluded objects in a cluttered scene.

4.1 Overview

The information gathering problems we are interested in are those in cluttered scenes where desirable sensor viewpoints are of nonuniform density. Motivating examples include object detection/classification for fruit picking [16] where fruit are occluded by leaves or each other and may be arranged in sparsely distributed clusters, highlighted in Figure 4.1. Other examples include condition monitoring for infrastructure inspection, where certain subassemblies or components must be detected and observed in detail.

In contrast to problem formulations such as map building and full scene reconstruction, we consider a formulation that requires targeted collection of information about unknown regions of interest (ROIs). The ROIs are small parts of the scene that function as a

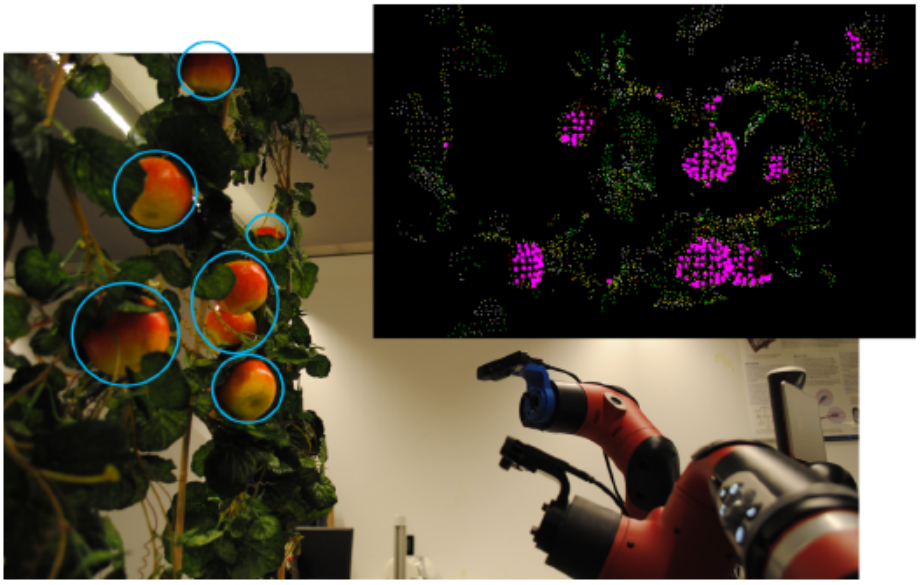


Figure 4.1: Searching for objects of interest, here apples, that are sparsely located and in some cases heavily occluded.

user-defined indicator which can be developed to support the perception task at hand. For example, here we define colour-based ROIs designed for object recognition from point cloud data; detecting an ROI subsequently focuses the collection of range data, and these two processes use different sensing modalities. We assume that successful observation of ROIs is heavily dependent on the viewpoints, and that full coverage of all viewpoints is not feasible. Thus an active perception approach is required. We refer to this problem as *targeted information gathering*.

Information gathering for systems with complex motion and sensor models faces several key challenges that have not yet been adequately addressed, thus preventing the wide-spread use of these techniques in practice, despite significant interest in developing new methods in recent years. It remains a challenge to effectively balance between exploration of an unknown scene, as we must do to detect ROIs, and targeted observation of key regions. Further, most techniques only consider simple motion models that are inappropriate for robots with nontrivial kinematics, such as high degree-of-freedom robot arms.

In this chapter we present a targeted information gathering algorithm for carrying out active perception, suitable for systems with complex motion and sensor models. Targeted information gathering of ROIs is guided by a viewpoint evaluation function that predicts

both how many ROIs may be discovered and how much they improve our model of previously discovered ROIs, quantified as volumetric information [85] which utilises ray tracing for the sensor model. The accurate cost of moving between viewpoints for high-dimensional arms is provided on-demand by HAP. MCTS is used to plan sequences of viewpoints for the robot that maximises the expected information gain (IG) over a long time horizon while satisfying budget constraints for the action costs.

4.1.1 Chapter Outline

The remainder of this chapter is organised as follows. Section 4.2 formally defines the problem including the robot, environment and observation models. Section 4.3 formally states the viewpoint planning problem in terms of IG to be solved by the proposed active perception framework. Section 4.4 describes the targeted information gathering volumetric information (VI) metric used within the IG objective. Section 4.5 describes the active perception framework including how the VI metric and HAP are integrated within the framework. Section 4.6 presents an apple harvesting experiment, validating the active perception framework.

4.2 Problem Formulation

We consider the problem of actively perceiving a set of regions of interest (ROIs) of a scene by a robot. The aim is to produce a point-cloud of the scene that reconstructs ROIs. The ROIs are initially unknown to the robots, however a bounding volume is assumed, and should be discovered then reconstructed as accurately as possible. The robot must plan a sequence of poses in order to make the best set of observations within a fixed travel budget.

4.2.1 Robot and Environment

For a robot r, let us denote a sequence of views $(v_1, v_2, ...)$ as τ. There is a finite set of candidate views, V where $V \subset \mathbb{R}^3 \times SO(3)$, such that each view $v \in V$ is a 6D

pose $t \in SE(3)$ reachable by the arm and with at least one collision free inverse kinematic solution. Here, V is equivalent to $\hat{T}$ as in the HAP formulation; however, a different notation is used for concreteness.

The workspace, W, is the 3D Euclidean workspace, $W = \mathbb{R}^3$. We define ROIs, W_{roi}, as the objects we are interested in reconstructing and to be a closed set of points in W. We assume W_{roi} to be unknown initially, however within some bounded box, $W_{box} \subset W$, such that $W_{roi} \subset W_{box}$.

Obstacles in the workspace, O_{env}, such as the robot's body and the ground, are known prior to reconstruction. The ROI bounding box, W_{box}, is additionally treated as an obstacle. The obstacle region, O, is then $O = W_{box} \cup O_{env}$. Let us define the configuration space, C, to be the set of all possible configurations of the robot. The robot is defined as a rigid body, $A \subset \mathbb{R}^3$, with a fixed base and robot arm configuration $q \in C$. The obstacle region is then: $C_{obs} = \{q \in C \mid A(q) \cap O \neq \emptyset\}$, where $A(q) \subset W$ is the space occupied by the robot and the sensor at configuration q. The free space region is then: $C_{free} = C \setminus C_{obs}$.

4.2.2 Observation Model

To represent W_{box} and W_{roi} an octree [192] volumetric representation is used for each, denoted X_{box} and X_{roi} respectively. An octree is made up of voxels, x, which have the boolean property of being occupied or unoccupied.

This work considers sensors that acquire a 3D scan of the environment. To simulate the visibility of the sensor, a set of rays, R_v, are cast with respect to the sensor pose. This is visualised in Figure 4.2. The rays are cast according to the following sensor parameters: field of view (FOV), the pixel resolution, the sensor view pose, and the depth of view. Let $X = X_{box} \cup X_{roi}$, then $X_v \subseteq X$ is the set of observed voxels for a view $v \in V$, where a voxel is observed if it is occupied and the first hit by a ray $r \in R_v$.

A view $v \in V$ can be evaluated by *information gain* (IG). *Volumetric information* (VI) is defined as the amount of information contained in a single voxel, x. IG is obtained by summing the VI over all observed voxels, X_v, for a view, v. The predicted IG, G_v, for a

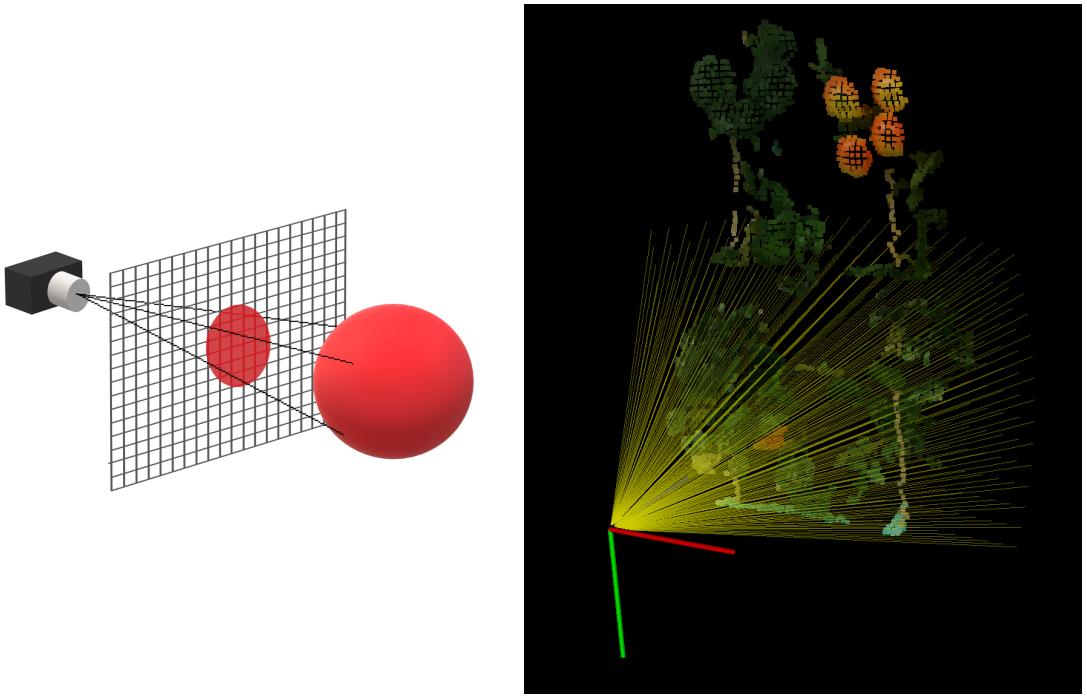


Figure 4.2: Visualisation of observation model which uses ray tracing to predict IG from a particular viewpoint.

single view $v \in V$ is:

$$G_v = \sum_{x \in X_v} I(x), \tag{4.1}$$

where $I(x)$ is a particular VI formulation (e.g., average entropy).

4.3 Problem Statement

A robot executing plan τ incurs an estimated execution cost, $h(\tau) \geq 0$ and given a cost budget $b > 0$ (e.g., time, energy). A view sequence τ is feasible if it meets the budget constraint such that $h(\tau) \leq b$.

We aim to solve the following optimisation problem:

Problem 4.1 (Viewpoint plannning)**.** *Given a set of candidate viewpoints V, find the optimal plan τ^* for the robot that maximises the information gain G_{V_τ} over the viewpoints:*

$$\tau^* = \arg\max_{\tau} G_{V_\tau}, \tag{4.2}$$

such that $h(\tau) \leq b$, *where* $V_\tau = \bigcup_{\forall v \in \tau} v$.

4.4 Viewpoint Evaluation

In this section we introduce a novel observation prediction model for targeted information gathering. Our method evaluates a set of viewpoints by measuring the amount of volumetric information over a set of ROIs in the environment. We introduce a quality criteria and penalty term to bias our selection of next view in order to balance between exploration of unobserved regions and exploitation of known ROIs to perform accurate reconstruction.

4.4.1 Surface Reconstruction

A set of 3D points called a *point cloud* $P \subset \mathbb{R}^3$ is used to store a partially reconstructed surface directly from the k-th sensor measurement P^k_{scan}. Over n-scans, the point cloud is

$$P = \bigcup_{n=1}^{k} P^n_{scan}. \tag{4.3}$$

ROIs are segmented from P via colour thresholding and once reconstruction is complete a GPIS-based object segmentation algorithm [155] detects apples. The algorithm is ideal for detecting partially reconstructed objects with high occlusion and when objects are clustered together since it uses 3D point information. The octree for the ROI points, X_{roi}, is re-initialised when the segmented ROI points $P_r \subset P$ are updated.

4.4.2 Volumetric Information Formulation

In our framework, the VI is extracted using an octree structure for ray casting. Additionally, the original point cloud P is retained which can be queried via the octree for richer information. This is important since the information of interest is sparse, whereas the voxel grid decreases the computation effort required for ray tracing.

A set of rays R_v casted from a view $v \in V$ returns a set of observed voxels X_v where X_v is the set of first voxels to intersect with each ray. The reason for only considering the first

intersection is to account for occlusions (e.g., leaves covering apples or apples occluding each other in clusters). To determine information I for an observed voxel $x \in X_v$, we use two VI formulations: exploration and targeted information gathering. Note that the sensor parameters used for generating R_v can be different for each VI formulation; this can be useful for reducing computation cost.

4.4.2.1 Exploration

A uniform planar point cloud is constructed over W_{box} and used to initialise the bounding box octree X_{box}. With a new view $v \in V$, the corresponding voxels $X_v \subset X_{box}$, are stored in a set M_e. The exploration information I_e for a voxel $x \in X_{box}$ is:

$$I_e(x) = \begin{cases} 1 & x \notin M_e \\ 0 & x \in M_e \end{cases}, \tag{4.4}$$

such that a voxel $x \in X_{box}$ can only contribute IG once.

It should be noted that for simplicity a non-probabilistic octree representation is used; however, this representation could be replaced with a probabilistic representation such as the octomap [149] data structure. As such, the exploration formulation could be replaced with any probabilistic approach, such as Kriegel's average entropy [147]. In fact, any prior could be used here if available.

4.4.2.2 Targeted information gathering

Volumetric information for a set of ROIs is evaluated by counting the number of ROI points where a hitting ray's angle of incidence relative to the surface normal, ψ, is within a threshold, ε. The incident angle-based evaluation is formally described as:

$$f(p) = \begin{cases} 1 & p \in P_r \wedge p \notin M_r \wedge \psi \leq \varepsilon \\ 0 & p \notin P_r \vee p \in M_r \end{cases}, \tag{4.5}$$

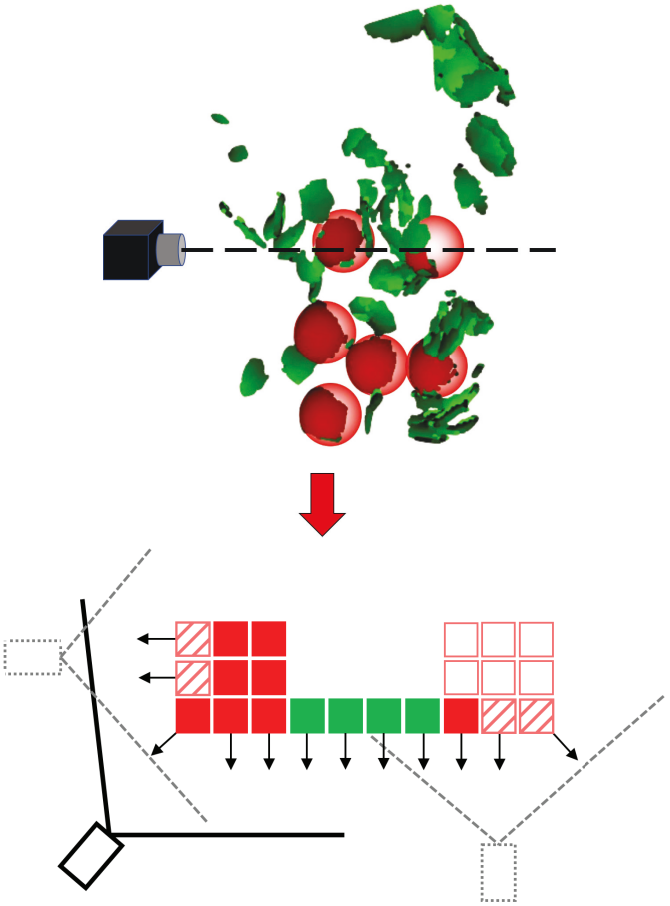


Figure 4.3: Visualisation of view evaluation in 2D: solid green are observed non-ROIs, red are observed ROIs that have contributed to the cumulative IG, striped red points are observed ROIs with potential IG, hollowed points are unobserved points and black arrows are point normals. The solid box is the current camera position and the dashed boxes are candidate views.

where M_r is similar to M_e except M_r tracks observed ROI points, $p \in P_r$, that have contributed IG, which are retrieved from their corresponding voxels.

This *quality criteria* encourages good visibility of ROIs with the intention to uncover partially observed areas. The ROI information I_r for a voxel $x \in X_{roi}$ is:

$$I_r(x) = \sum_{\forall p \in x} f(p). \tag{4.6}$$

To encourage ROI exploration, a penalty term, $\sigma(p)$, is applied to $f(p)$ if contributed ROI points are within proximity of the candidate point p:

$$\sigma(p) = 1 - \tanh\left(\frac{k(p)}{k_{max}}\right), \tag{4.7}$$

where $k(p)$ is the number of neighbouring points in M_r within a fixed sphere around p and k_{max} is a user specified maximum number of neighbouring points. This function is a monotonically increasing penalty with a horizontal asymptote, k_{max}. The revised ROI VI formulation is:

$$I_r(x) = \sum_{\forall p \in x} f(p)\sigma(p). \tag{4.8}$$

This VI formulation is motivated in Figure 4.3, where two candidate views (dashed boxes) are being evaluated. Without the penalty term $\sigma(p)$, these views have the same IG. However, there is a higher chance of observing more ROIs from the right view because there are less number of ROI points that have already been considered. Another effect is to prevent the robot from perseverating on a cluster of ROIs and instead opt for exploring other parts of the scene once a reasonable reconstruction has been achieved.

Furthermore, in certain applications sensor pixel footprint can vary greatly based on distance of objects to the sensor. In these settings a weighting on the ROI VI based on distance can be added to account for this.

4.4.2.3 Multi-objective information gain

Given the proposed criteria for VI, the IG in (4.1) for a view G_v is re-written in the form of a weighted sum:

$$G_v = \sum_{\forall x \in X_v} (1-\beta) I_e(x) + \beta I_r(x), \tag{4.9}$$

where $\beta \in [0, 1]$ is a user defined weighting.

4.5 Active Perception Framework

In this section, we propose a non-myopic active perception planning algorithm for the robot to plan its action with respect to the perception model detailed in Section 4.4. We assume a discretisation of the the search space is given. The algorithm is based on MCTS which uses HAP for estimating the motion costs of the robot arm.

4.5.1 Monte Carlo Tree Search

MCTS is a general heuristic search algorithm for solving decision processes such as board games [193]. It is a non-myopic random sampling approach which makes it suitable for information gathering problems. The algorithm works by expanding a tree based on random sampling of the action space, in our case the actions are sensor viewpoints. In doing so, MCTS realises the most promising sequence of actions and uses this to select its next action. In our context promising means maximising IG. This occurs in rounds and consists of four steps:

- *Selection:* Start from a root node (current pose of the robot) and choose a sequence of child nodes until a leaf has been reached.
- *Expansion:* Unless executing this leaf node action exhausts the robot's budget, b, expand the node with another valid action.
- *Simulation:* Complete a random playout, or rollout, from this newly created node until budget b is exhausted.
- *Backpropagation:* Use the result of the rollout, the total IG, to update the score of the nodes in the path from the root node to the newly create node.

The above process is run for a maximum number of iterations, the most promising action is taken and then the whole process repeats. An important decision is selecting a sequence of child nodes in the *selection* step above. While many variants of MCTS exist for solving this problem [162], a popular choice is called Upper Confidence Bound 1 applied to trees (UCT) [194]. UCT effectively balances between exploration of new sequences of actions and exploitation of existing promising ones and provides theoretical guarantees for a polynomial bound on regret, see Appendix B.2 for more details. However, even when using UCT the algorithm generally requires a large amount of rollouts to reach a reasonable solution.

In the following sections it is described how the targeted information gathering VI and HAP are integrated into the MCTS algorithm to form our active perception framework. Importantly, these components help to overcome the computational burden of MCTS and retain real-time performance and a faithful description of the behaviour of the system.

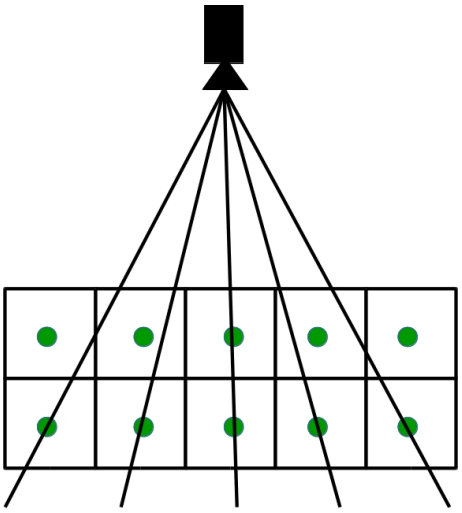
4.5.2 Reward Function

In MCTS, during the rollout a reward function is used to evaluate how effective a particular action was. This is exactly the viewpoint evaluation function defined in (4.9). As the tree is expanded, the observed VI sets M_e and M_r are incrementally updated based on predicted observations. During a rollout a temporary copy of these are made to predict future IG which is then backpropagated to child node scores. It was found that a small discount factor helped reduce variance and improve detection rate as this encourages MCTS to not commit to actions too far into the future.

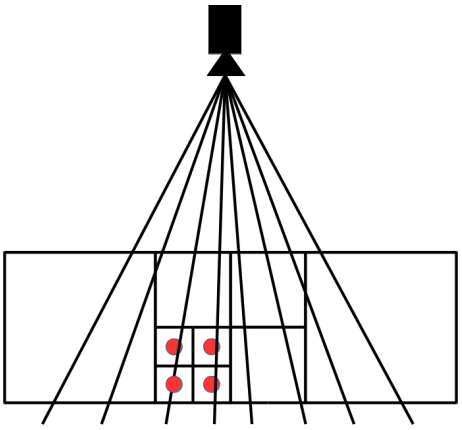
In addition to UCT, the multi-objective IG balances between exploration of the scene and targeted observations of ROIs. Hence, the idea is that they work in harmony to achieve effective reconstruction of objects of interest in a small amount of MCTS iterations. Reducing the amount of needed MCTS iterations is important given the computationally expensive nature of ray tracing which is needed for every reward function evaluation.

Furthermore, ROIs are a noisy yet fast to compute indicator of where objects of interest are. Once reconstruction is complete, these ROIs are then passed to the GPIS segmentation algorithm for refinement. This paradigm allows for fast reward estimation whilst producing a targeted dataset which is beneficial for downstream processes.

This computational burden is further reduced by using two separate octrees X_{box} and X_{roi} and set of rays R_{v_e} and R_{v_r}. The reason is that X_{roi} can afford to have a higher resolution when the ROIs are sparse, as is our case, and X_{box} can have a lower resolution which is usually sufficient for exploration. The case is the same for the pixel resolutions of R_{v_e} and R_{v_r}. This is visualised in Figure 4.4. Furthermore, the original ROI point cloud which X_{roi} is constructed over is retained and richer information can be accessed via the octree with no extra computational cost on the ray tracing operation. As such, our method enables effective use of a complex sensor model with MCTS while retaining fast computational performance.



(a) X_{box}



(b) X_{roi}

Figure 4.4: Two separate octrees (shown here as quadtrees for simplicity) used for the exploration and targeted information VIs. X_{box} has a more uniform structure where as X_{roi} has a more sparse structure, resulting in larger voxels filling the empty space.

4.5.2.1 MCTS rollout with targeted information gathering VI

Algorithm 4.1 elaborates on how the targeted information gathering VI is incorporated into the rollout of MCTS. The procedure initially creates a temporary copy of the observed VI sets, denoted $\hat{M}_e$ and $\hat{M}_r$. Then a k-d tree is constructed from $\hat{M}_r$ which is used for computing the penalty term $\sigma(p)$ in (4.7). Then, each view v in the sampled rollout τ is iterated over and the IG is accumulated from both the exploration and targeted information gathering VI metrics I_e and I_r. Notice that these VI metrics are computed from separate environment data structures and even the set of rays used for ray tracing

Algorithm 4.1 MCTS rollout with targeted information gathering VI

Input: ROI point cloud $\boldsymbol{P_r}$, Octrees $\boldsymbol{X_{box}}$ and $\boldsymbol{X_{roi}}$, and observed VI sets $\boldsymbol{M_e}$ and $\boldsymbol{M_r}$

Output: Multi-objective IG $\boldsymbol{G_v}$

1: Create temporary copies $\hat{M}_e$ and $\hat{M}_r$
2: Construct k-d tree from $\hat{M}_r$
3: **for each** v in sampled τ **do**
4: $X_v \leftarrow X_{box}$`.raytrace(`R_{v_e}`)`
5: **for each** x in X_v **do**
6: Accumulate I_e and update $\hat{M}_e$
7: **end for**
8: $X_v \leftarrow X_{roi}$`.raytrace(`R_{v_r}`)`
9: **for each** x in X_v **do**
10: $\{p\} \subset P_r \leftarrow x$`.get_points()`
11: **for each** p in $\{p\}$ **do**
12: **if** $\psi \leq \varepsilon$ **and** $p \notin \hat{M}_r$ **then**
13: $k(p) \leftarrow$ neighbouring points in k-d tree
14: Accumulate I_r with $\sigma(p)$ and update $\hat{M}_r$
15: **end if**
16: **end for**
17: **end for**
18: Update k-d tree with $\hat{M}_r$
19: Apply discount factor
20: Accumulate G_v
21: **end for**

can be constructed with different sensor parameters (see lines 4 and 8).

A subtle detail is that $\sigma(p)$ only considers neighbouring points in $\hat{M}_r$ that were added from previous view evaluations in the sampled sequence. This is reflected in line 19 where the k-d tree is updated with $\hat{M}_r$ only after accumulating I_r for that view. Finally, a discount factor is applied, which increases for views later in the sampled sequence, and the multi-objective IG G_v is accumulated. The rollout procedure returns the final accumulated G_v for backpropagation in the MCTS algorithm.

4.5.3 Computing Realistic Trajectory Costs for Budgeting

In our implementation of MCTS a rollout terminates once a given execution time budget, b, is exhausted. This means that an estimate of the execution time of each action, in our case to move the sensor on the end-effector of the robot arm to a viewpoint, needs to be

computed for every action in the rollout and for every rollout. Hence, a fast estimate of h is needed.

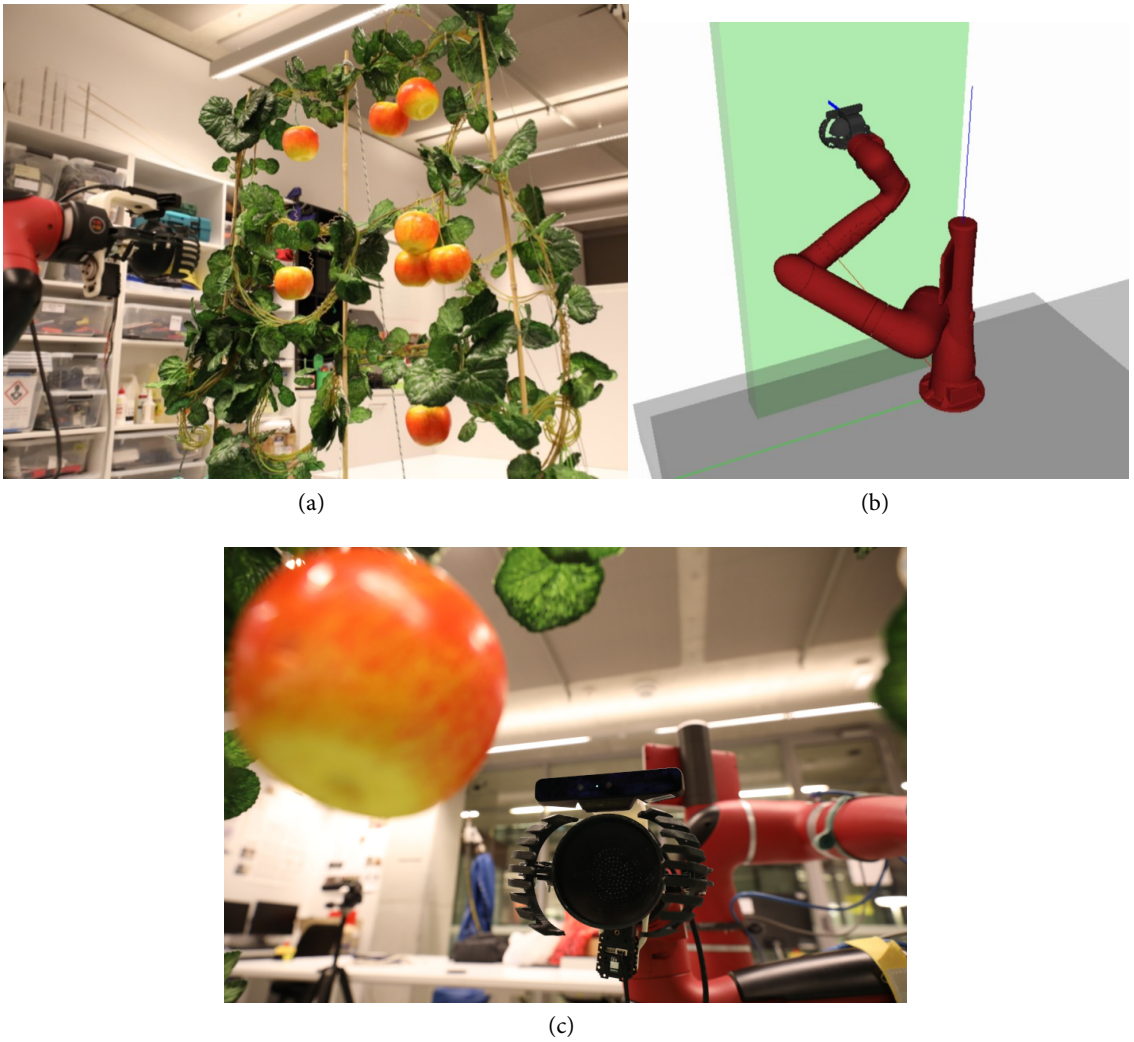
In order to address the inherent computational complexity, we employ HAP to compute these costs for MCTS. Note that the HAs found by HAP use the usual d_C trajectory metric and h is computed from the trajectory libraries, which could be some other metric such as time. In doing so, we enable faithful description of the complex manipulator motion model within the active perception framework.

4.6 Apple Harvesting

To validate the active perception framework, an experiment is setup where a robot arm with an eye-in-hand sensor has to find apples on a trellis and then pick them. Here we apply the active perception framework described above to find the apples and then utilise HAP to sequence and pick the apples. Note that we do not evaluate the targeted information gathering VI metric here, rather we reserve this for the next chapter where the active perception framework is generalised for multiple robots.

4.6.1 Experimental Setup

For the experiments a seven-DOF Rethink Robotics Sawyer is mounted on a custom made static base with adjustable height. A Realsense SR300 is used as an eye-in-hand sensor to provide RGB-D data. The gripper used is an in-house custom made design that uses a single screw actuator to open and close its jaws around the apple. The gripper actuation consists of two stages, first a linear stroke to push any obstructions away from the centre of the gripper and then finally a closing motion to secure the apple. The design intends that the jaws act as human fingers, encompassing the apple regardless of its position and angle of approach. Furthermore, contact between the apple and the gripper is reduced, mitigating the risk of bruising the apple. The experiments were conducted with an artificial apple trellis with realistic appearance. The experimental setup can be viewed in Figure 4.5.



(a) (b)

(c)

Figure 4.5: Experimental setup for apple harvesting experiments. (a) A Sawyer robot arm with eye-in-hand sensor and novel gripper end-effector assembly and realistic artificial apple trellis, (b) simulated environment and (c) close up of the end-effector assembly.

A time budget, b, of 5 seconds was given for the active perception phase. Ten experiments were carried out with the Sawyer arm which attempted to harvest 8 apples from the trellis. For each experiment a unique apple configuration is manually arranged on the trellis, see Figure A.3 for images of all the configurations used. The aim was to provide a diverse range of scenarios such as clusters of apples with varying sizes and occlusions.

The experimental setup is an effective test bed since it emulates sources of uncertainty that the proposed control and planning method should be able to deal with in the real world. This includes errors from multi-view fusion of point clouds and apples moving around as a consequence of other apples being detached.

4.6.2 Implementation Details

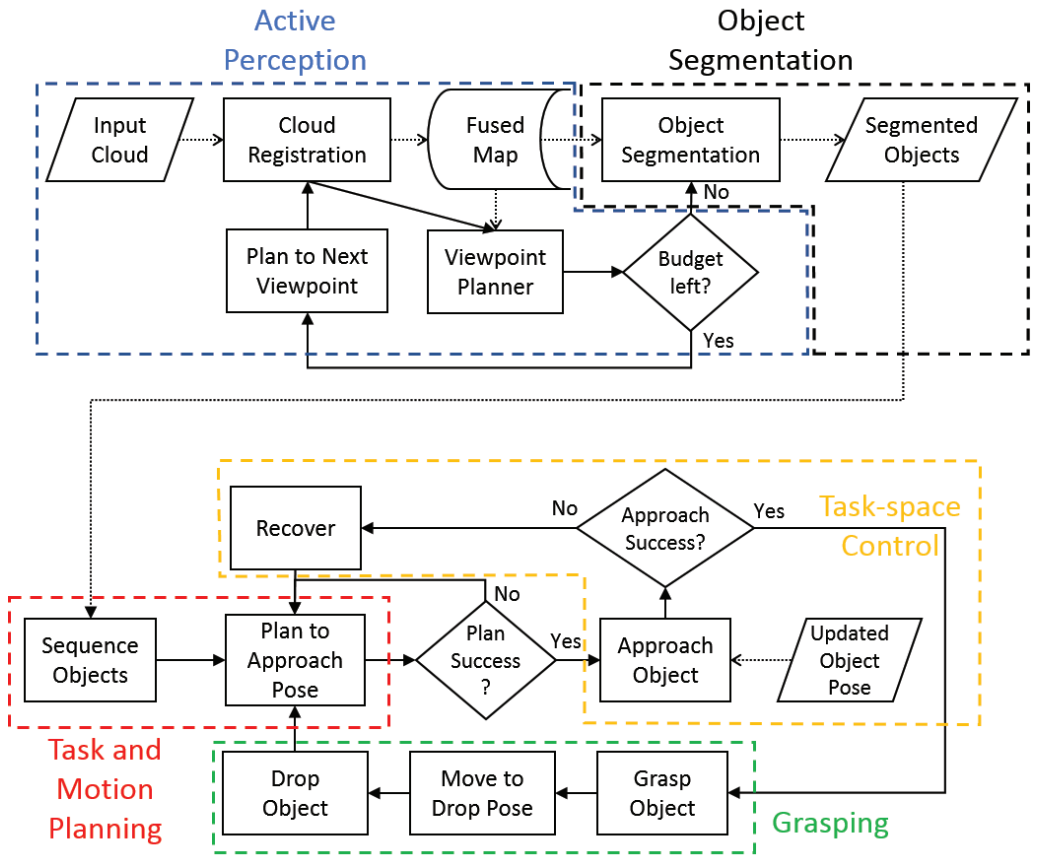


Figure 4.6: Apple harvesting execution pipeline and subsystem layout. There are five main subsystems: active perception, object segmentation, task and motion planning, task-space control and grasping.

This experiment combines several algorithmic components developed so far in this thesis. As such, how these components are integrated and implementation specific details are described below.

4.6.2.1 Software architecture

The software architecture for the apple harvester is shown in Figure 4.6. While the implementation details for the components are described here, this is a general architecture and could be applied to other object grasping applications. There are five main subsystems:

- *Active Perception:* This is the implementation of the active perception framework described in Section 4.5. MCTS is the viewpoint planner. It computes a sequence of viewpoints and if the budget hasn't been exhausted then HAP is used to move the robot arm to the next viewpoint. The eye-in-hand sensor takes a scan of the scene and then aligns and fuses the point cloud with its local map. This process is repeated until the budget is exhausted.

- *Object Segmentation:* Once the budget is exhausted the object segmentation procedure executes. To segment the apples, the ROIs are first extracted from the final fused map. These are processed by a GPIS algorithm [155] which is a more refined segmentation algorithm and uses the 3D point information for filtering noise and separating apples in clusters. GPIS provides uncertainty estimates for each point based on how closely they fit a spherical prior, however, they are not used in this experiment.

- *Task and Motion Planning:* Once the apples are segmented, HAP is used to sequence the order in which to pick them. To pick them, HAP is used to plan and place the gripper at an approach pose in front of the next apple in the sequence.

- *Task-space Control:* From the approach pose, task-space control is used to move the gripper towards to the apple whilst updating its position using sensor feedback from the eye-in-hand sensor. To update the apple's position, the GPIS algorithm continuously runs and the new estimate of the position is calculated using a Kalman filter. If the approach succeeds then the grasp routine is executed, otherwise a recovery procedure is executed where the arm reverses its trajectory back to the approach pose. This is repeated until all apples are picked.

- *Grasping:* The grasping procedure is straight forward: the gripper is commanded to close its jaws in an open loop manner. To detach the apple the arm simply reverses its trajectory back to the approach pose, which is additionally the drop pose. To drop the apple the jaws are opened again.

4.6.2.2 VI hyper-parameter selection

The targeted information gathering VI hyper-parameters were chosen by incrementally changing them and observing which ones resulted in higher number of correctly detected apples. β was varied between 0.1 and 0.9 with 0.5 performing best. The rollout discount factor was varied between 0.8 and 1.0 with 0.9 performing best. The k_{max} parameter was varied between 10 and 100 ROI points with 20 performing best which was intuitively the approximate the number of ROI points required for the segmentation algorithm to detect an apple. ε was varied between $\pm\boldsymbol{\pi}$ radians and 2 radians was found to be optimal. The ball radius for considering neighbouring points in the calculation of the penalty term (4.7) was 0.15m and the number of MCTS iterations was 200.

The minimum octree leaf resolution used for construction of X_{box} and X_{roi} is 0.07cm and 0.03cm, respectively; note smaller here means higher overall resolution. The reduced pixel resolution for the camera model used for ray tracing X_{box} is 15×20 and 24×32 for X_{roi}. The FOV of the camera model is additionally reduced by one third and the depth of the rays limited to 1.0m to further mitigate the impact of occlusions.

4.6.2.3 Approach Pose Selection

The approach pose is chosen in a similar fashion to the task-space control experiments for the HAP evaluation in Section 3.7.4. The main difference is that here the approach target is the centre of the apple found by the GPIS segmentation algorithm. The orientation of the approach pose is always normal to the trellis plane.

4.6.2.4 HAP implementation details

Again, the discretisation strategy is similar to Section 3.7.4 where we assume that the apples lie roughly in some vertical 2D plane and generate a set of uniform 6D poses normal to this plane and additionally a set of slightly angled poses for looking around occlusions. The set of viewpoints used are shown in Figure 4.7. The edge costs are the same as those used in the task-space control experiments for the HAP evaluation in Section 3.7.4 to avoid joint limits and singularities.

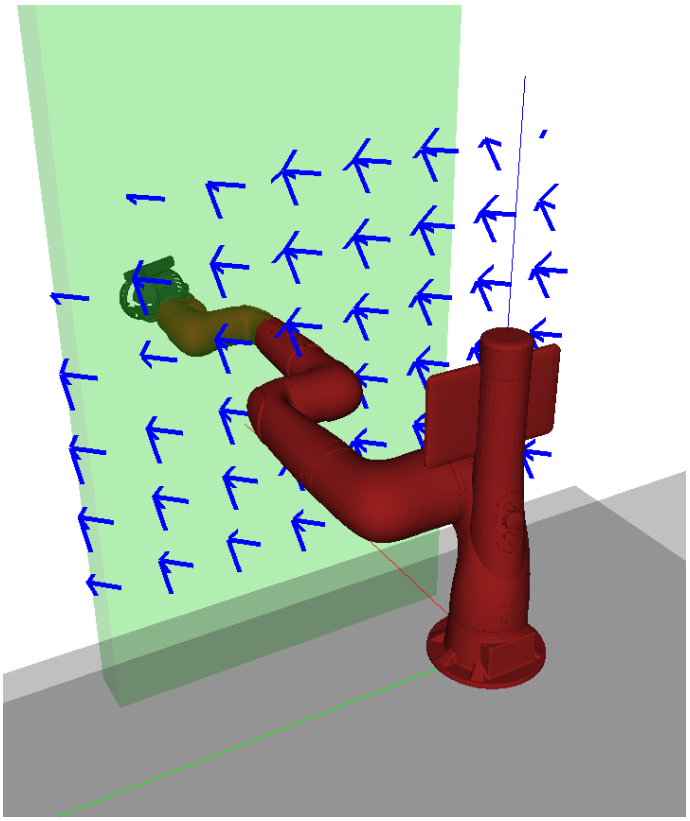


Figure 4.7: Discretised viewpoints computed by HAP that are used by the active perception framework to search over.

4.6.2.5 Approaching the apples

Once the gripper has reached the approach pose, the approach routine moves the gripper linearly towards the goal. Differently to the UR5, the Sawyer is a redundant arm meaning there may be an infinite number of joint velocities that achieve the same end effector velocity. In this case, the pseudoinverse solution in (3.17) produces a minimum-norm velocity solution. However, we may wish to exploit the arm's redundancy to improve some constraint based criteria such as avoiding joint limits [195], self-collisions [196] or kinematic singularity [190].

This can be achieved by modifying (3.17) to the following form [197]:

$$\dot{q_d} = \mathbf{J}^{+} \begin{bmatrix} \dot{e_d} \\ \dot{\alpha_d} \end{bmatrix} + (\mathbf{I} - \mathbf{J}^{+}\mathbf{J})\dot{q}_0 \tag{4.10}$$

where $\dot{e_d}$ and $\dot{\alpha_d}$ are now with respect to the centre and orientation of the gripper. The second term in the equation projects an arbitrary joint velocity vector $\dot{q_0}$ into the null-space of the Jacobian. Intuitively, it exploits the redundancy to generate internal motions that change the manipulator configuration without changing the end-effector pose. The choice of $\dot{q_0}$ is explained in detail in Appendix B.1.

Since the repulsive potential contribution of $\dot{q_0}$ is mapped into the null-space of the Jacobian, it remains possible for the manipulator to violate constraints, such as remaining within its joint limits. As such, HAP remains useful for avoiding violation of these constraints, such as joint limits, since we can encourage HAs to find configuration mappings that, for example, start further away from joint limits.

During the approach, we use the eye-in-hand camera configuration to update the apple location by running the GPIS algorithm continuously and passing the new estimate through a Kalman filter. If more than one apple is in view of the camera then the apple closest to the original position estimate is used for the update.

We additionally detect if the configuration approaches a joint limit or falls below a given manipulability threshold, in which case we terminate the approach. If the robot does not encounter a servoing failure, it executes the grasp once at the goal. Once complete, we reverse the approach trajectory to return the gripper to the approach pose.

4.6.3 Hardware Results

The viewpoint sequence computed by the active perception component for one of the trellis configurations is shown in Figure 4.8. Notice the arm taking complimentary views of the trellis, balancing exploration of unknown regions and targeted observations of ROIs whilst taking into consideration occlusions caused by apple clusters. This is highlighted in Figure 4.8g and Figure 4.8h where additional scans of the partially observed apple clusters are taken from downward facing views to provide more 3D information. This is particularly important for effective performance of the GPIS algorithm which utilises this 3D information to discern individual apples in clusters. The active perception generally took no more than a few seconds to compute the next view action.

(a)

(b)

(c)

(d)

(e)

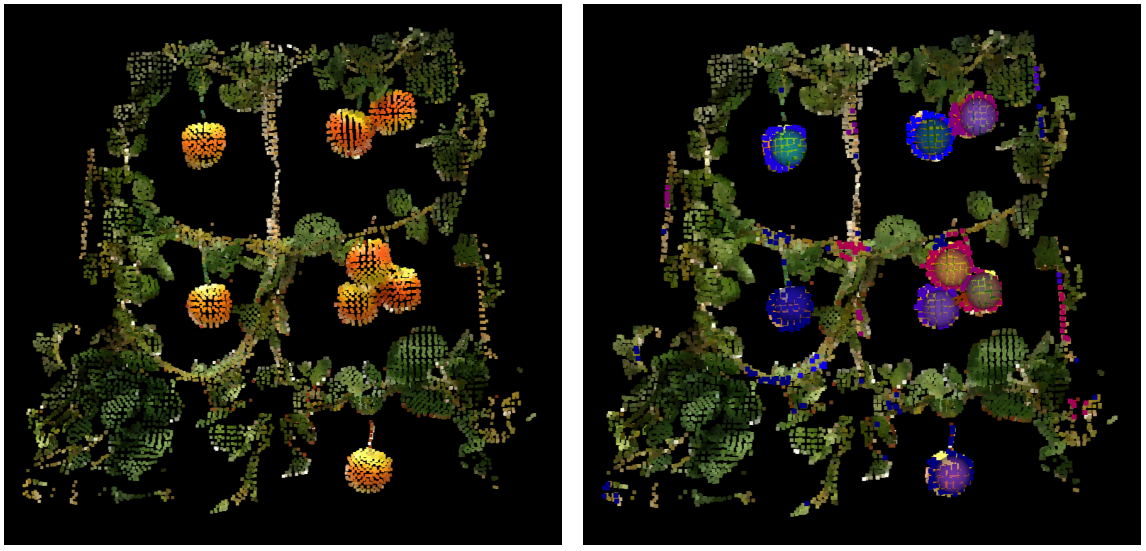
(f)

(g)

(h)

(i)

Figure 4.8: An example sequence of views computed by the active perception component in the apple harvesting experiment. (a)-(i) shows the arm taking complimentary views of the trellis, balancing exploration of unknown regions and targeted observations of ROIs whilst taking into consideration occlusions caused by apple clusters.



(a) (b)

Figure 4.9: Resulting reconstruction (a) and (b) segmented point cloud with GPIS entropy overlay (blue means low entropy and red means high entropy).

The final reconstructed point cloud is shown in Figure 4.9a. Figure 4.9b shows the GPIS segmentation algorithm output. As can be seen, the active perception framework produces a sufficient 3D reconstruction of the ROIs given the short time budget. This is evidenced by the output of the GPIS algorithm which correctly detects all the apples and additionally is able to resolve clusters effectively.

An example sequence of the arm grasping an apple is shown in Figure 4.10. Figure 4.10a shows the arm commencing its approach towards the apple. As can be seen in the top right of the figure, the GPIS algorithm is running on the local view of the camera to update the position of the apple. Furthermore, this demonstrates a situation where two apples are in view and the system is able to resolve the correct target. Figure 4.10b and Figure 4.10c demonstrate the intended behaviour of the gripper design where as it approaches the target it passively pushes the other apples out of the way before executing the grasp.

The results of the experiments are found in Tables 4.1 and 4.2. The last column shows the success rate of the experiments. The proposed system could, on average, successfully harvest 88.75% of all apples in the experiments.

As shown in Table 4.2, the main causes of failures are due to the following:

(a)

(b)

(c)

(d)

(e)

Figure 4.10: An example grasp sequence from the apple harvesting experiment. (a)-(e) shows the following in order: arm commencing approach and updating target position, arm at target (notice the gripper passively pushing the other apple out of the way), gripper closing to grasp the apple, arm moving to drop position and gripper opening to release apple.

- *Misalignment*: the gripper is not aligned properly with the target apple, missing it or closing the jaws on it.
- *Gripper Obstruction*: the gripper jaws get caught onto the trellis structure and the gripper stops the closing or opening of the jaws.
- *Failed Detachment*: the gripper grasps the target apple; however, it slips out of gripper jaws while detaching from the trellis.

Table 4.1: Experimental results for apple grasping.

Experiment	Plan Time (s)	Plan Execution (s)	Approach Time (s)	Success (%)
1	0.07	3.11	11.42	100.00
2	0.07	2.82	11.46	75.00
3	0.05	3.06	11.42	100.00
4	0.09	3.41	11.45	100.00
5	0.06	3.05	11.52	75.00
6	0.22	3.80	11.45	87.50
7	0.05	3.15	11.42	100.00
8	0.04	2.74	11.47	75.00
9	0.07	3.10	11.39	75.00
10	0.10	3.65	11.52	100.00
Average	0.08	3.19	11.45	88.75

Table 4.2: Classification of hardware experiment failures.

Experiment	Misaligned	Gripper Obstructed	Failed Detachment	Poor Prioritisation
1	0	0	0	0
2	1	0	0	1
3	0	0	0	0
4	0	0	0	0
5	0	0	1	1
6	0	0	0	1
7	0	0	0	0
8	1	1	0	0
9	0	0	1	1
10	0	0	0	0
Total	2	1	2	4

- *Poor Prioritisation*: the gripper hits an apple from the front or near the target apple, pushing it out of the gripper's jaws or an apple is unintentionally detached whilst another target apple is grasped.

High success rates suggest robustness to a diverse range of apple configurations. This success was aided by the robust active perception framework which effectively reconstructed the ROIs, here apples, within a short time budget. This is evidenced by the downstream segmentation algorithm correctly detecting all the apples in the experiments. Moreover, there was no plan, joint limit or low manipulability failures.

The failures in Table 4.2 can be largely attributed to the gripper and its interaction with the artificial trellis. Although the environment is realistic, it is not fully representative, i.e.

real apples are heavier and more rigidly attached. This may help prevent some failures resulting from poor prioritisation. Further, we believe that with a modest amount of gripper adjustments many of the failures could be mitigated.

4.7 Summary

This chapter presented a framework for targeted information gathering in cluttered, nonuniform scenes where object perception is assisted by indicative ROIs. A novel evaluation method was proposed for viewpoints and efficiently computed an information-maximal sequence of viewpoints for the robot. The experimental results show that the proposed framework is capable of detecting occluded and sparsely located objects of interest using novel quality criteria and penalty terms. Furthermore, it was shown that using a complex sensor and motion model within this framework was possible. In the next chapter we extend the framework's formulation to a decentralised setting where sensor data can be exchanged between multiple agents in order to collaboratively gather information.

Chapter 5

Decentralised Coordination Framework

This chapter introduces the decentralised coordination framework which scales the active perception framework in Chapter 4 to multiple robots. This framework overcomes the computational complexity of planning for multiple robots with complex motion and sensor models whilst dealing with the real world challenges of deploying a decentralised robot system. The framework is evaluated in a hardware experiment consisting of multiple high-DOF robot manipulators with eye-in-hand sensors searching for objects of interest in a cluttered environment. Empirical results evaluate the effectiveness of our approach.

5.1 Overview

Scaling active perception problems to multiple robots allows for more efficient operation, for example robots can take complementary views and cover a greater portion of an environment in a shorter time. The drawback of this is that coordination is required, making the algorithms more complex, and hence the computational requirements increase since the search space scales exponentially in the number of robots.

Our goal is to develop a generalised form of coordinated planning that can be applied to a variety of these types of active perception problems, and additionally allow for the

use of multiple robots. Most prior work has focused on single-robot settings, and it remains a challenge to efficiently scale these techniques up for multiple robots, particularly decentralised multi-robot systems. It is believed that overcoming these limitations is key to achieving an effective general active perception system that can deal with the most difficult problem settings.

In this chapter we present the decentralised coordination algorithm for carrying out multi-robot active perception. Decentralised coordination between multiple robot arms is achieved using Decentralised Monte Carlo tree search (Dec-MCTS) [82]. Dec-MCTS is used to plan sequences of viewpoints for the team of robots that maximises the joint expected IG over a long time horizon while satisfying budget constraints for the action costs. The coordinated behaviour of the robots is aided by periodically communicating probability distributions over action sequences, and the point clouds from observations.

5.1.1 Chapter Outline

The remainder of this chapter is organised as follows. Section 5.2 reformulates the active perception problem for a team of robots and then modifies the viewpoint planning problem for the decentralised setting in Section 5.3. Section 5.4 describes the decentralised coordination framework, including how Dec-MCTS is utilised, how the targeted information gathering VI accounts for multiple robots, and the overall high-level system execution. Section 5.5 presents an extensive evaluation of the targeted information gathering metric both in simulation and hardware using multiple robot arms with eye-in-hand sensors searching for objects of interest. Preliminary experiments are presented in Section 5.6 where the robot needs to manipulate obstructions in order to observe ROIs, demonstrating the flexibility of the decentralised coordination framework.

5.2 Robot Team

We now consider the problem of actively perceiving a set of regions of interest (ROIs) of a scene by a team of coordinated robots. The robots must coordinate their planned

sequences of poses in order to make the best set of observations within a fixed travel budget.

A team of robots is denoted $\mathcal{R} = \{r^1, r^2, ..., r^N\}$, where N is the number of robots. For each robot $r^i \in \mathcal{R}$, let us denote a sequence of views $(v_1^i, v_2^i, ...)$ as τ^i. The joint plan for the team of robots is denoted $\Pi = \{\tau^1, \tau^2, ..., \tau^N\}$.

5.3 Problem Statement

A robot executing plan τ^i incurs a known execution cost, $h(\tau^i) \geq 0$ and cost budget $b^i > 0$ (e.g., time, energy). A joint-set of view sequences Π is feasible if all robots meet the budget constraints such that $h(\tau^i) \leq b^i, \forall r^i \in \mathcal{R}$.

We aim to solve the following optimisation problem:

Problem 5.1 (Decentralised viewpoint planning)**.** *Given a set of candidate viewpoints V and a team of robots $\mathcal{R}$, find the optimal joint plan Π^* for the team of robots that maximises the overall information gain G_{V_Π} over the viewpoints:*

$$\Pi^* = \arg\max_{\Pi} G_{V_\Pi}, \tag{5.1}$$

such that $h(\tau^i) \leq b^i$ for all robots $r^i \in \mathcal{R}$, where $V_\Pi = \bigcup_{\forall v \in \Pi} v$.

This problem is to be solved in a decentralised manner where each robot considers its own plan τ^{i*} while communicating with others to coordinate. It is important to note that the joint IG is not the sum of individual IGs over robots since we want to avoid redundant IG by the team of robots.

5.4 Decentralised Coordinated Planning

In this section, we propose a decentralised planning algorithm for the robots to plan their actions with respect to the perception model detailed in Section 4.4. The algorithm is based on the Dec-MCTS [82] which we apply to our problem by combining it with HAP

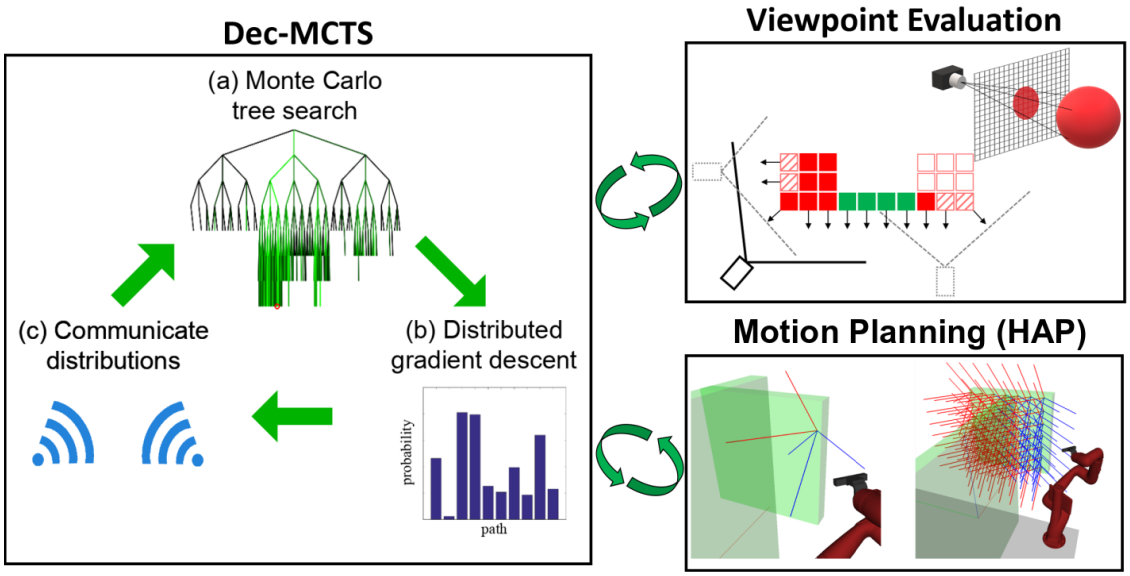


FIGURE 5.1: Decentralised planning framework. *Left*: Dec-MCTS computes sequences of viewpoints. *Top-right* A viewpoint evaluation function used within MCTS as optimisation objective. *Bottom-right*: HAP computes accurate time costs to move which is used to limit the depth of the tree search in MCTS.

and the targeted information gathering VI described in Chapter 4. We describe these components and their interaction below. A high-level view of the decentralised planning framework is visualised in Figure 5.1.

5.4.1 Decentralised Monte Carlo Tree Search

The team of robots optimise their sequences of viewpoints using the Dec-MCTS planning algorithm. In Dec-MCTS, each robot asynchronously cycles between three phases:

1. Perform a tree search over its own action space.
2. Compress the search tree into a probability distribution.
3. Broadcast these distributions to other robots.

When new messages are received from other robots, this information is incorporated in future rounds of the tree search. These phases continue until the robot is ready to execute its decision. Replanning is performed after new observations. For node selection Dec-MCTS uses a novel variant of UCT called Discounted UCT (D-UCT), see Appendix B.3,

which accounts for changing reward distributions as a consequence of incorporating other agents' plans which are non-stationary.

The main difference we use in this implementation to the original algorithm in [82] is a more efficient compression of the search tree for communication. Periodically, the estimated n best paths according to MCTS are saved along with an associated probability distribution. The probabilities are proportional to the normalised expected reward for each path, plus a fixed constant. This formulation does not provide the theoretical guarantees discussed in [82, Prop. 1] yet is much more efficient in practice since it does not require computing expensive multiple integrals.

5.4.2 Integration with Active Perception Framework

Algorithm 5.1 Dec-MCTS rollout with targeted information gathering VI

Input: ROI point cloud $\boldsymbol{P_r^i}$, Octrees $\boldsymbol{X_{box}}$ and $\boldsymbol{X_{roi}}$, and observed VI sets $\boldsymbol{M_e}$ and $\boldsymbol{M_r}$

Output: Joint multi-objective IG $\boldsymbol{G_{v_\Pi}}$

Create temporary copies $\hat{M}_e$ and $\hat{M}_r$
Construct k-d tree from $\hat{M}_r$
for each r^i in $\mathcal{R}_{sorted}$ **do**
  for each v in sampled τ^i **do**
    $X_v \leftarrow X_{box}$`.raycast(`R_{v_e}`)`
    for each x in X_v **do**
      Accumulate I_e and update $\hat{M}_e$
    end for
    $X_v \leftarrow X_{roi}$`.raycast(`R_{v_r}`)`
    for each x in X_v **do**
      $\{p\} \subset P_r^i \leftarrow x$`.get_points()`
      for each p in $\{p\}$ **do**
        if $\psi \leq \varepsilon$ **and** $p \notin \hat{M}_r$ **then**
          $k(p) \leftarrow$ neighbouring points in k-d tree
          Accumulate I_r with $\sigma(p)$ and update $\hat{M}_r$
        end if
      end for
    end for
    Update k-d tree with $\hat{M}_r$
    Apply discount factor
    Accumulate G_{v_Π}
  end for
end for

The first phase in Dec-MCTS uses MCTS to plan over the single robot's action space while considering the predicted actions of other robots. This planning is performed to optimise with respect to the objective function (5.1) which is now the overall joint IG of all robots.

The action execution costs are estimated, again, using HAP for time budgeting. Differently, collected sensor data is now broadcasted and shared between all agents. Each robot collects these individual scans and fuses them into their own local map, P^i, which significantly reduces the amount of data being communicated.

As the robot executes view actions the observed VI sets M_e and M_r are incrementally updated as in the single robot case, however, now the other robots' executed viewpoints are included in the update. To encourage complementary observations, predicted observations of other robots are additionally incorporated by sampling paths from the communicated probability distributions. This is performed during an individual robot's rollout in MCTS similar to Algorithm 4.1; however, now there is an additional outer for loop where other robots' sampled action sequences are considered first and the updated temporary copies of M_e and M_r carry over into the next robot's sequence until eventually its own. This is shown in Algorithm 5.1 for concreteness where $\mathcal{R}_{sorted}$ is the team of robots sorted such that the current robot's sampled path is iterated over last.

5.4.3 System Execution

Algorithm 5.2 Decentralised Coordination Framework

Input: Roadmap of feasible actions $\boldsymbol{V}$ and budget $\boldsymbol{b^i}$
Output: Reconstructed map $\boldsymbol{P^i}$

1: **while** budget for robot r^i remaining **do**
2: Receive sensor data from other robots
3: Update P^i, M_e and M_r
4: Run Dec-MCTS (Alg. 5.3)
5: Execute next action in computed sequence
6: Capture sensor data
7: Update P^i, M_e and M_r
8: Transmit sensor data to other robots
9: **end while**

The flow of execution for a single robot in the team is summarised in Algorithm 5.2. First the robot receives any new sensor data from the other robots and updates its locally

Algorithm 5.3 Dec-MCTS Overview

Input: Remaining budget for robot $\boldsymbol{r^i}$
Output: Sequence of actions $\boldsymbol{\tau^{i*}}$

```
Initialise MCTS tree
while computation budget not met do
    Select most promising set of action sequences
    for N iterations do
        Grow MCTS tree
        Update action sequence distribution
        Communicate compressed tree
        Receive compressed trees
    end for
end while
```

reconstructed map and observed VI sets M_e and M_r. Then an iteration of Dec-MCTS is run as is shown in Algorithm 5.3.

During the MCTS phase a search tree is grown over the robot's action space using D-UCT. This tree growth is performed while considering the probability distributions over the other robots' plans. Periodically the robot's most promising action sequences are updated based on the tree search. During the communication phase the robot communicates its tree in a compressed form as a probability distribution. If the robot receives a new distribution from any of the other robots then in the next iteration the most promising action sequences are optimised while considering this new information.

Once a computation budget is met then Dec-MCTS terminates and returns the action sequence with the highest probability. The robot executes the first action in this sequence, then captures new sensor data, transmits this sensor data, and updates its local map and observed VI sets M_e and M_r. This is repeated until the robot's budget b^i is exhausted.

5.4.4 Problem Complexity Analysis

Theorem 5.1 (Complexity of decentralised viewpoint planning problem)**.** *Given number of robots N and set of discretised viewpoints V, the number of all possible solutions for* Π *is:*

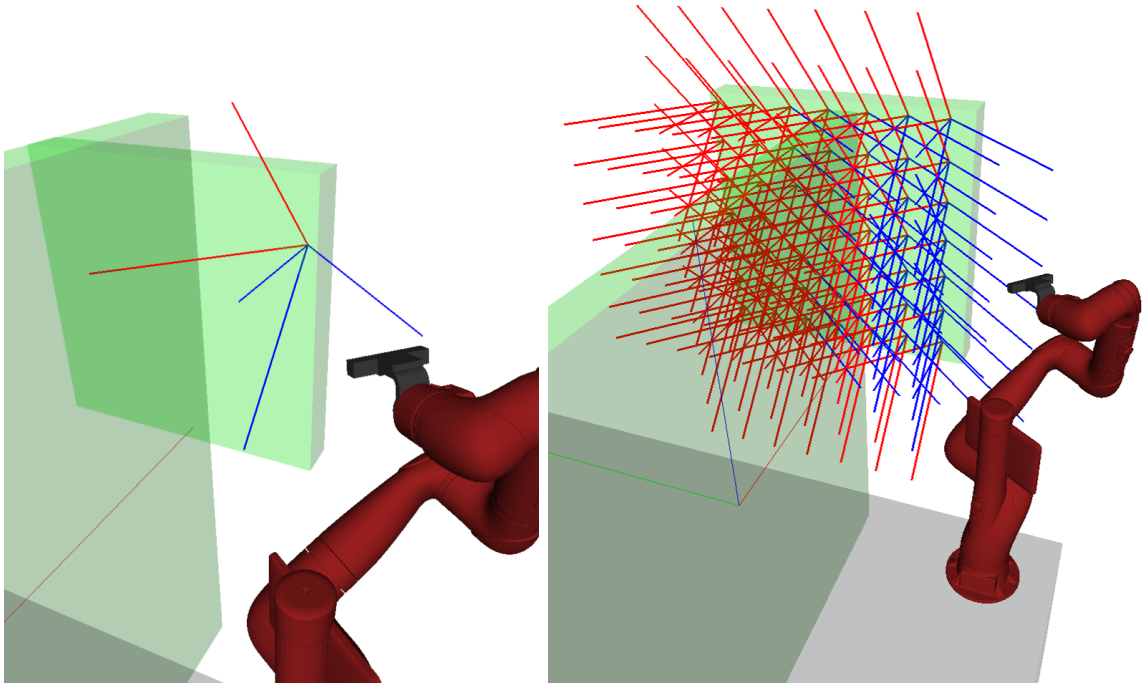
$$\frac{(|V|+1)!}{(|V|-N-2)!},$$

FIGURE 5.2: Apple recognition scenario. *Top-left*: Multiple robot arms with RGB-D sensors attached and overlapping views of a mock-up apple trellis. *Bottom-left* The workspace of the robot arms are virtually separated. *Right*: The pointclouds captured by the arms are broadcasted and fused together.

which is factorial in the number of robots N.

Proof. Given a sequence of viewpoints τ^i for i-th robot, the major computational bottleneck is in finding the corresponding configuration $q \in Q(v)$ for each viewpoint $v \in \tau^i$ in order to compute the cost $h(\tau^i)$. Instead of considering every possible pair of configurations for a given trajectory between viewpoints, HAP finds a single optimal IK solution $q^* \in Q(v)$ at each viewpoint $v \in V$.

Unlike existing multi-robot planning approaches that are centralised and exhaustive, our framework is online and anytime. This is important since the number of possible solutions is intractably large. Without loss of generality, $\tau^i \cup \tau^j = \emptyset$ for all $i \neq j$ in Π^*. In the worst case, $|\bigcup_{\tau \in \Pi^*} \tau| = |V|$ when b^i is sufficiently large for all robots. □



(a) (b)

Figure 5.3: Roadmap construction: Each vector represents a candidate view with a sensor origin and look-at point, a red vectors indicate unreachable poses. (a) View point construction and (b) resulting set.

5.5 Searching for Objects with Multiple Robot Manipulators

We demonstrate the performance and behaviour of the decentralised coordination algorithm with extensive simulated experiments and a realistic hardware implementation. We consider the scenario illustrated in Figure 5.2, where a team of two robotic arms coordinate their actions to best find and observe a set of apples on a trellis. The apples, which are the ROIs, are partially occluded by leaves and thus require a judicious selection of viewpoints to be sufficiently observed and modelled. This scenario represents the necessary information gathering component of precision agriculture tasks, where the gathered information is later used to guide intervention, such as picking or spraying fruit [1, 198].

(a) (b)

Figure 5.4: (a) Trellis used for simulations that has four clearly-visible apples and two apples occluded by leaves and (b) for hardware experiments with five sparsely located apples.

5.5.1 Experimental Setup

We evaluated our algorithm with extensive simulation experiments, using a dataset with real sensor data, and in hardware that required real-time decision making by the robots. The task was for two robot arms to cooperatively find and reconstruct apples on a mock-up trellis. The apple trellis configuration in Figure 5.4a was used for simulated experiments and the trellis in Figure 5.4b was used for the hardware experiments. This reconstruction is utilised by the GPIS segmentation algorithm to classify and localise apples. Both robots begin with no knowledge of the apple trellis other than it's bounding box. An experimental instance ends when both arms have exhausted their given budget and segmented their final fused map of the scene.

We used two Rethink Robotics Sawyer arms with Intel SR300 RGB-D sensors shown in Figure 5.2. Each robot had two dedicated Intel i7 NUC mini PCs, one for motion planning and arm control and the other for perception and planning. Here, perception includes: RGB-D image capture, mapping, data fusion, IG computation, and segmentation. The

computation time for each decision varied; however, it was generally less than a few seconds. The robots communicate their beliefs and observations over a wirelessly connected robot operating system (ROS) network [199]. A virtual wall is placed between the robots so their configuration spaces do not overlap, as shown in Figure 5.3. However, importantly, their observation regions do overlap. The same hardware was used to collect the dataset for the simulations.

The robots were given 5 seconds of execution time budget each. The roadmap used is visualised in Figure 5.3 which is made up of 128 unique feasible viewpoints. Hyperparameters were incrementally tuned and chosen as with the apple harvesting experiments in Section 4.6.2.2.

5.5.2 Simulation Results

For the simulated experiments, real sensor data was collected and used in the loop with simulated robots. Sample viewpoints from the dataset can be seen in Figure 5.5. Notice that some apples are only clearly visible from certain viewpoints and completely occluded from others. To evaluate the performance of our method we compare it with variations of itself and with other state-of-the-art IG metrics.

The simulated experiments were run 20 times per variant. In Figure 5.6b, random's detection rate was highly variable whereas all variants of our method performed consistently. Interestingly, in Figure 5.6a the difference was more pronounced when only considering the bottom two apples in Figure 5.4a. Without the penalty term, the robot exhausts its budget reconstructing the top cluster of apples. Without the quality factor, views that were occluded by leaves were not mitigated. Without communicating beliefs, there was significant overlap in views between the robots. Further, Dec-MCTS without beliefs achieved a higher exploration percentage at the expense of observing less ROI points. Including the probabilistic beliefs produced the best performance, particularly when using our proposed distribution weighting.

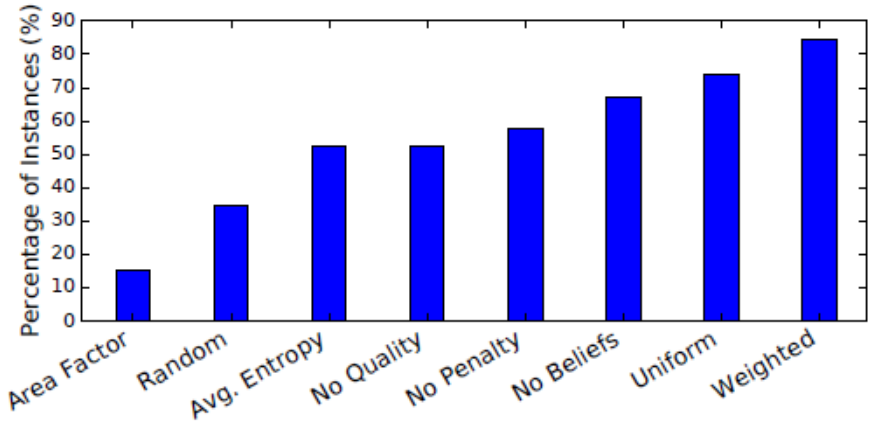
Average entropy and area factor [147, 152] exhibited overly exploitative view choices, attempting to reconstruct the trellis completely. Area factor was affected worst due to the

Figure 5.5: Samples viewpoints from dataset used for simluated experiments

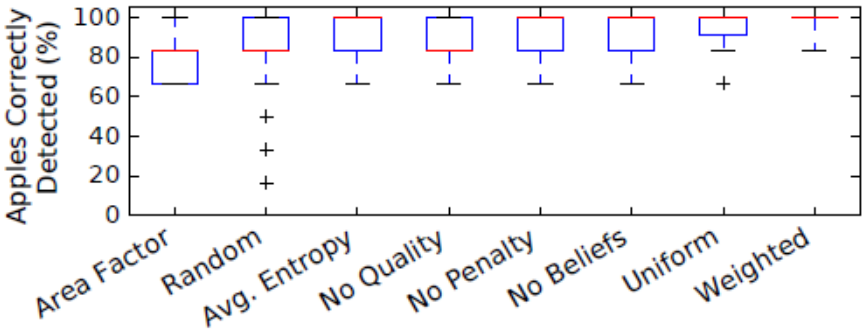
high amount of resulting occplanes [152] from the discontinuous trellis structure. Further, both formulations' IG queries are computationally expensive and not deemed feasible for real time execution without significant performance degradation.

5.5.3 Hardware Results

To evaluate the performance in hardware, ten instances of each VI formulation were run on a more difficult apple trellis configuration with sparsely-located and highly-occluded apples, visualised in Figure 5.4b. As can be observed in Figures 5.8a and 5.8b, our approach achieved a higher exploration percentage and discovered more ROI points overall than random, following the trend seen in the simulation results. Furthermore, our method on



(a) Number of instances both bottom apples in Fig. 5.4a were detected.

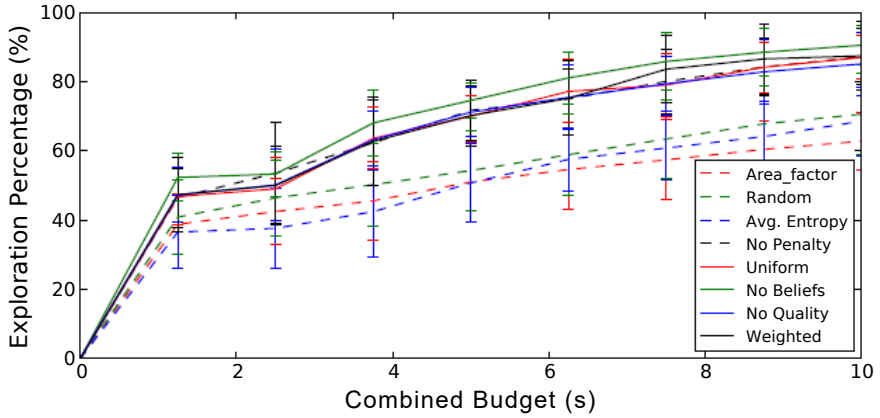


(b) Percentage of correctly detected apples.

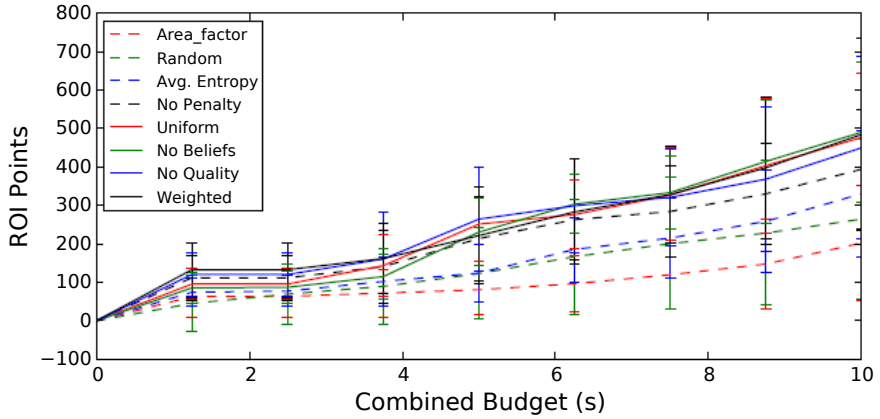
Figure 5.6: Apple detection.

average found 100% of the apples with a minimum of 80%, while random on average found 80% with a minimum of 40%.

We believe our formulation outperformed random due to several factors. One key insight is that the improvement in the number of apples detected was large in comparison to the increase in detected ROIs. This indicates that although random resulted in a relatively high number ROI points, our formulation arguably found ROI points that were more important to the classification of the apples, namely they traded off exploiting any one apple in favour for revealing more of others. This is attributed to the penalty term, σ, in (4.7), which encourages ROI exploration.



(A) Percentage of environment explored.

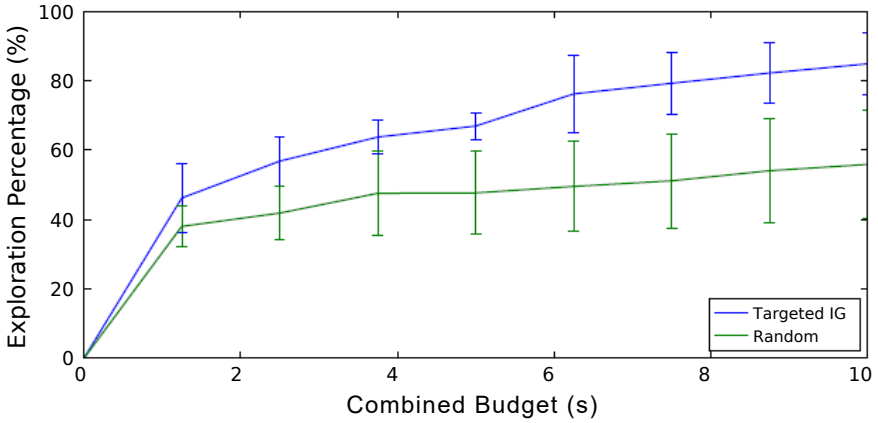


(b) ROI points plotted against combined travel time.

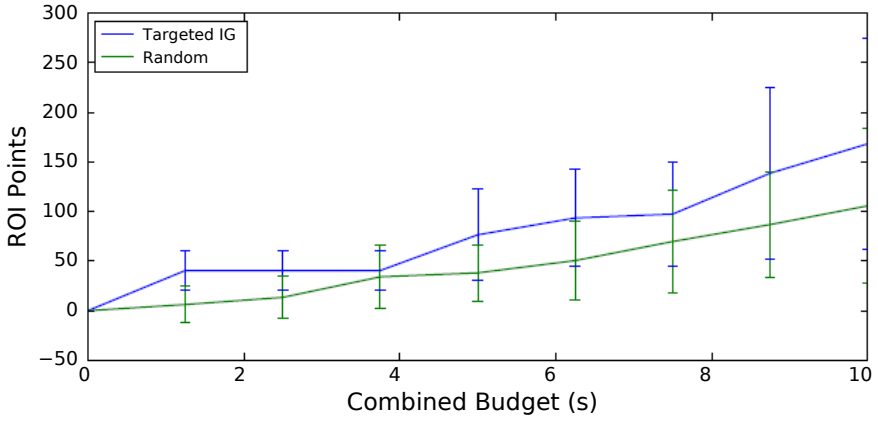
Figure 5.7: Simulated experimental results.

5.6 Unoccluding Objects Via Manipulation

The decentralised coordination framework is designed to be general and can facilitate behaviour beyond simply scanning the environment from particular viewpoints. One interesting problem in cluttered environments is that objects may be occluded in such a way that no camera viewpoint will reveal them. Real world examples of this are shown in Figure 5.9. Hence, there may be a need to manipulate parts of the environment in order to facilitate observation of those occluded objects. We refer to this problem as manipulation for perception. We demonstrate this idea using an illustrative experiment which will be detailed in the following sections.



(A) Percentage of environment explored.



(b) ROI points plotted against combined travel time.

Figure 5.8: Hardware experimental results.

5.6.1 Experimental Setup

In order to demonstrate this manipulation for perception idea the following experiment is setup where, similar to previous experiments, the robot arm with an eye-in-hand sensor has to locate apples on an artificial trellis. However, this time there are two obstructions blocking the sensor from observing some objects of interest from particular views. The experimental setup is shown in Figure 5.10 with the two obstructions highlighted. Since we are not claiming to solve this problem and given this is an illustrative experiment the obstruction locations and motions to move the obstructions are pre-programmed. Furthermore, the roadmap used by MCTS is simplified to be less dense and only contain views orthogonal to the trellis compared to that used in Section 5.5.

(a)

(b)

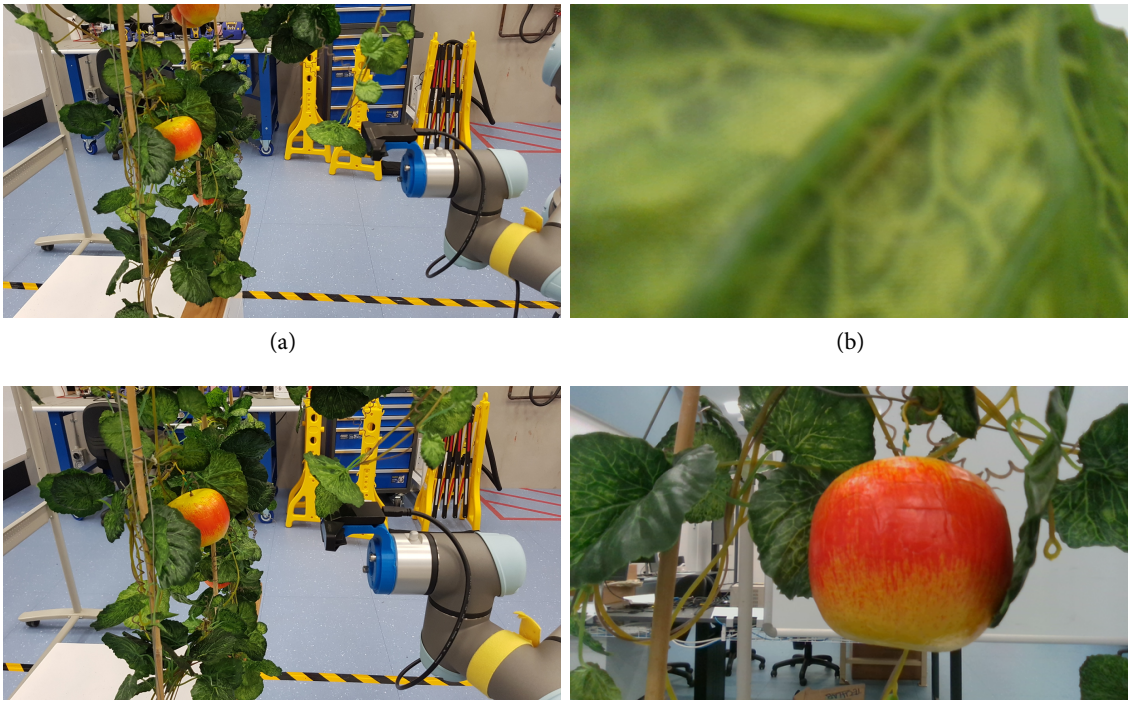
Figure 5.9: Examples of obstructions in a real apple orchard (Photos taken in Bilpin, NSW Australia).

Figure 5.10: Experimental setup for unoccluding objects via manipulation.

Two types of manipulation actions are used, a "pushing" and "poking" of obstructions. The intention here is to show that a diverse set of manipulation actions could be considered and if these could be identified and have a reward associated with them then they could seamlessly be integrated into the active perception pipeline. For example, these could be added to the action space of MCTS which would then determine which action is best and when in the sequence to perform them, just like with regular viewpoints. This modification would occur between lines 4 and 21 of Algorithm 5.2 where manipulation actions would also be sampled in addition to taking viewpoints and their reward computed then accumulated. The pushing and poking actions are described as follows.

5.6.1.1 Poking obstructions

For smaller obstructions with low resistance one could use a simple poking action. This is shown in Figure 5.11 where a single leaf dangling by a branch is covering the camera. In this case, the arm can simply poke the camera through the leaf to reveal the apple behind it.



(a) (b)

(c) (d)

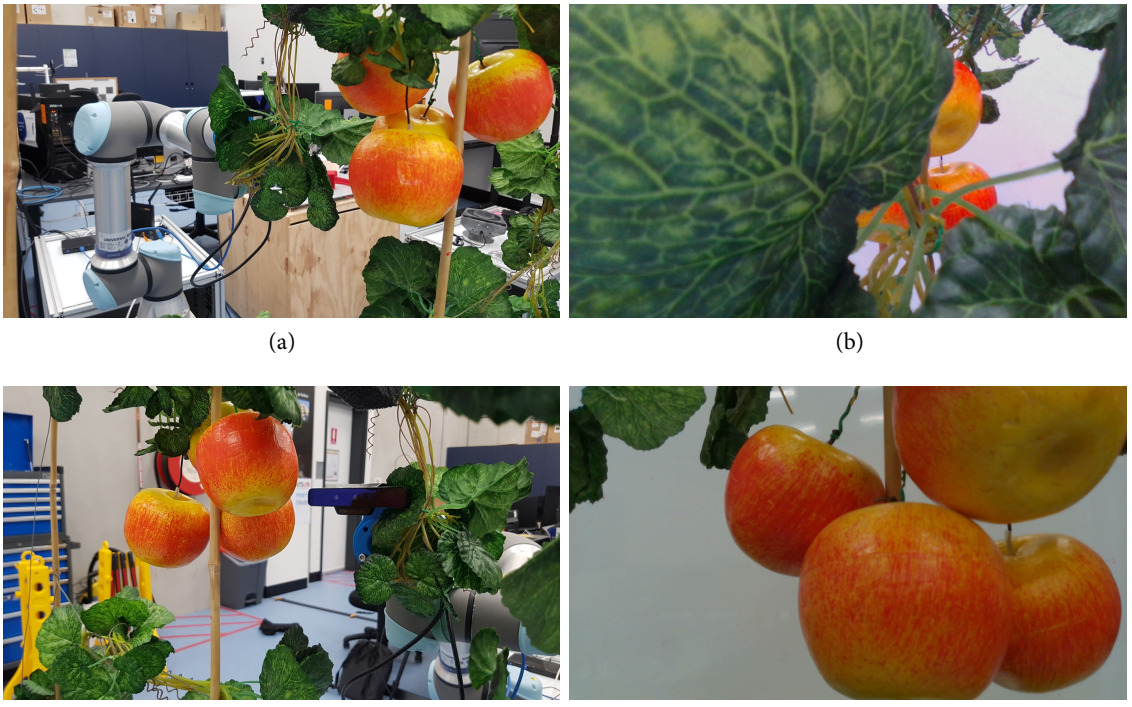
Figure 5.11: Poke obstruction example. (a) Obstructed sensor, (b) obstructed camera view, (c) unobstructed sensor and (d) unobstructed camera view.

5.6.1.2 Pushing obstructions

The poking action is simple and potentially more robust when used in the correct situation, however may not work for larger obstructions. An example of such an obstruction is shown in Figure 5.12 where the arm has to place its end effector to the side of the obstruction and then push the entire mass to the side in order to reveal objects behind it.

5.6.2 Hardware Demonstration

The experiment in this section is executed using the same active perception pipeline as in earlier experiments with the difference being that when the sensor is obstructed the arm executes the pre-programmed manipulation action based on which obstruction it is in front of, which additionally is known prior. An obstruction is detected when there is large enough points in the point cloud within a threshold distance from the sensor origin.



(a) (b)

(c) (d)

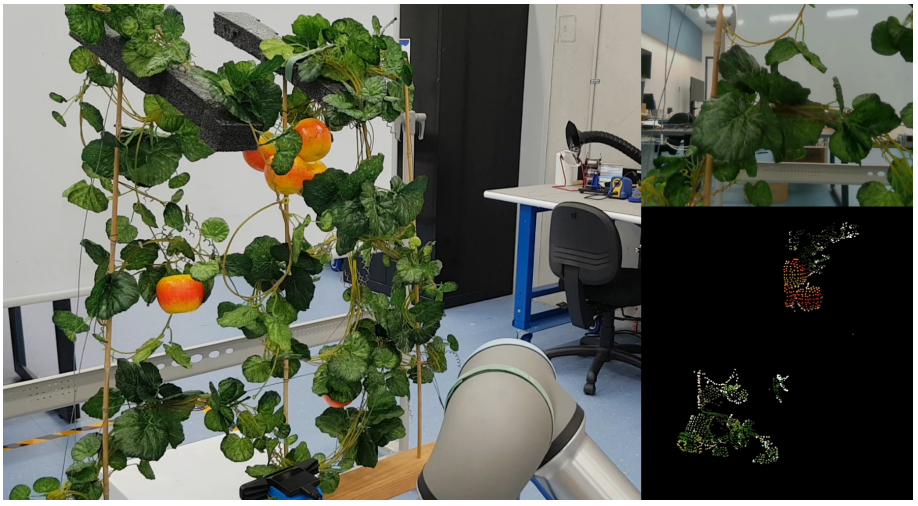
Figure 5.12: Push obstruction example. (a) Obstructed sensor, (b) obstructed camera view, (c) unobstructed sensor and (d) unobstructed camera view.

The resulting sequence is shown in Figure 5.13 (a)-(h) with inset camera view and point cloud reconstruction. The final point cloud reconstruction with segmented apples is shown in Figure 5.14.

5.7 Summary

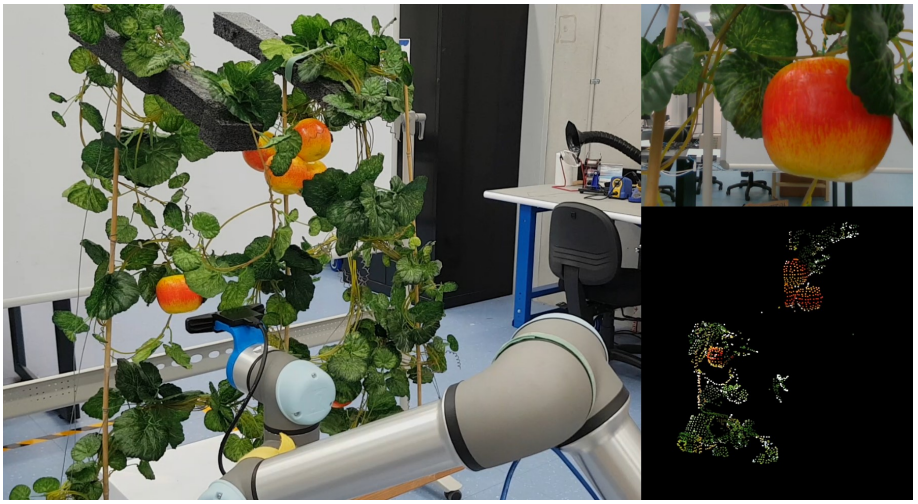
This chapter presented an extension of the active perception framework from Chapter 4 which efficiently computes an information-maximal set of plans for a team of robots. It does this in a decentralised manner while retaining the ability to utilise a complex sensor and motion model. The proposed VI metric was compared with several other state-of-the-art metrics and it was shown that the proposed method was able to consistently find all of the objects of interest within a low number of viewpoint evaluations. Additionally a manipulation from perception experiment was carried out which showcased the generalisability of the proposed framework.

(a)

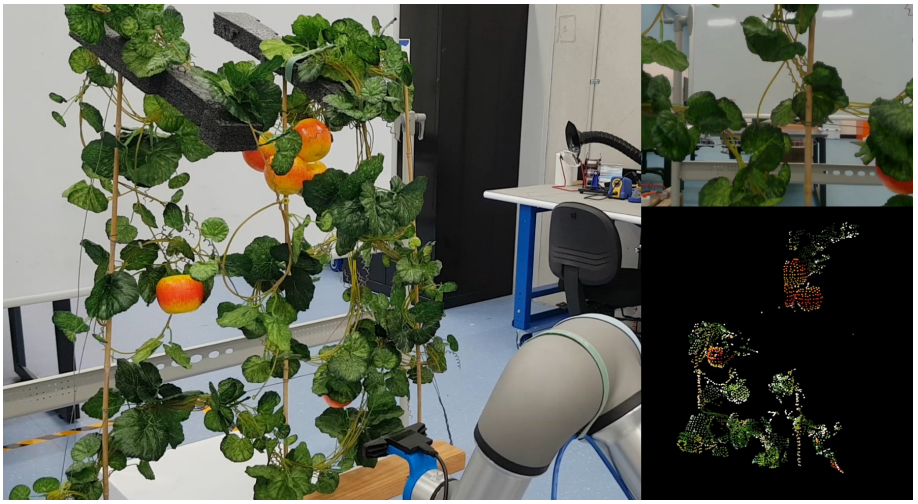


(b)

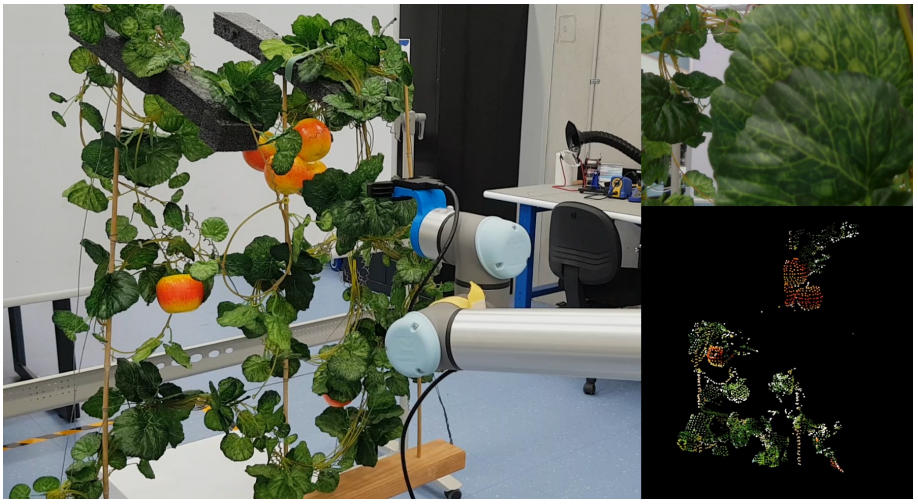
(c)



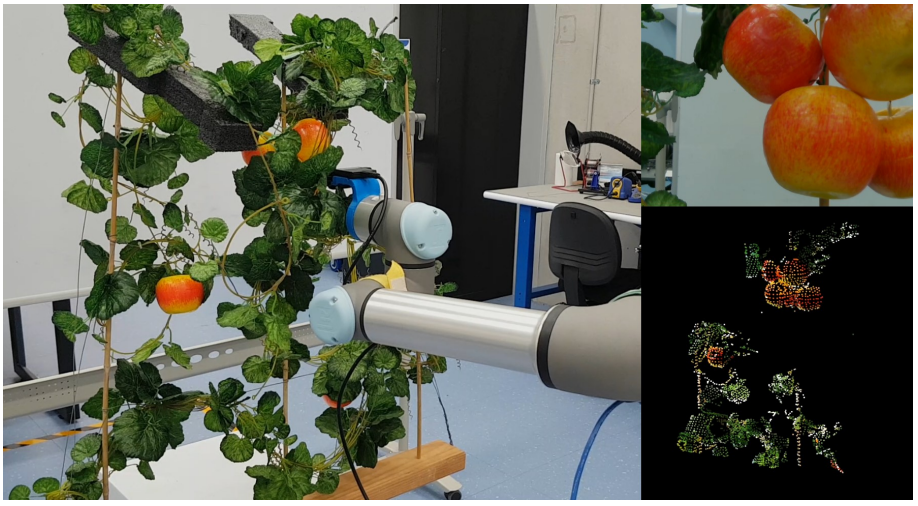
(d)



(e)



(f)



(g)

(h)

Figure 5.13: Hardware demonstration of manipulator unoccluding objects for active perception. (a)-(h) shows the sequence of viewpoints taken with camera view on the top right and reconstructed point cloud on the bottom right. Note the arm poking the obstruction in (d) and pushing the obstruction in (g).

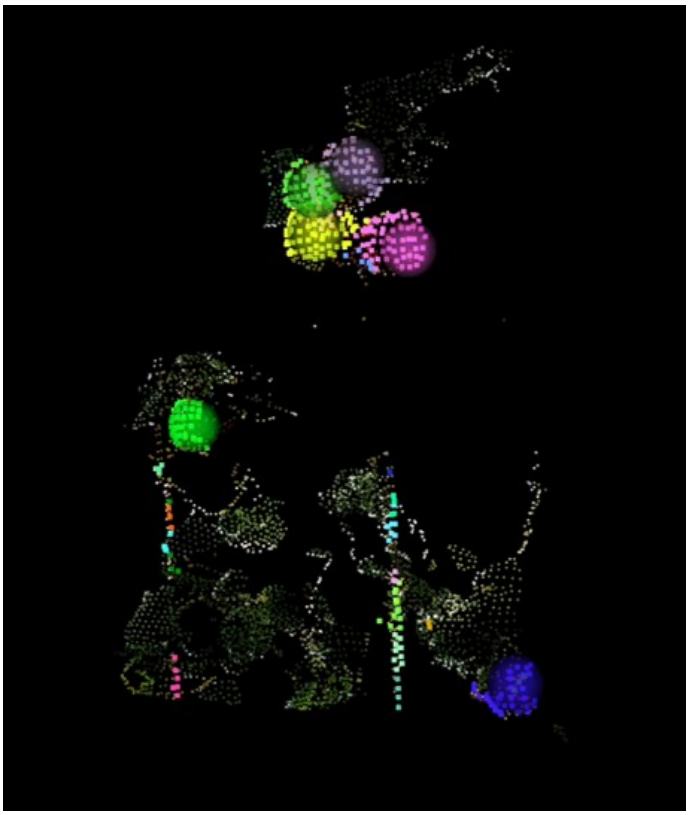


Figure 5.14: Final reconstructed point cloud with segmented apples. The ROIs include apples and other parts of the trellis such as the trellis structure, which is incorrect. This is the trade off of using ROIs; they are very fast to compute, colour thresholding in this example, however they are noisy. As such, they act as an indicator function for data that will be useful for some downstream task such as a more refined yet slower segmentation algorithm, for example the GPIS algorithm we use.

Chapter 6

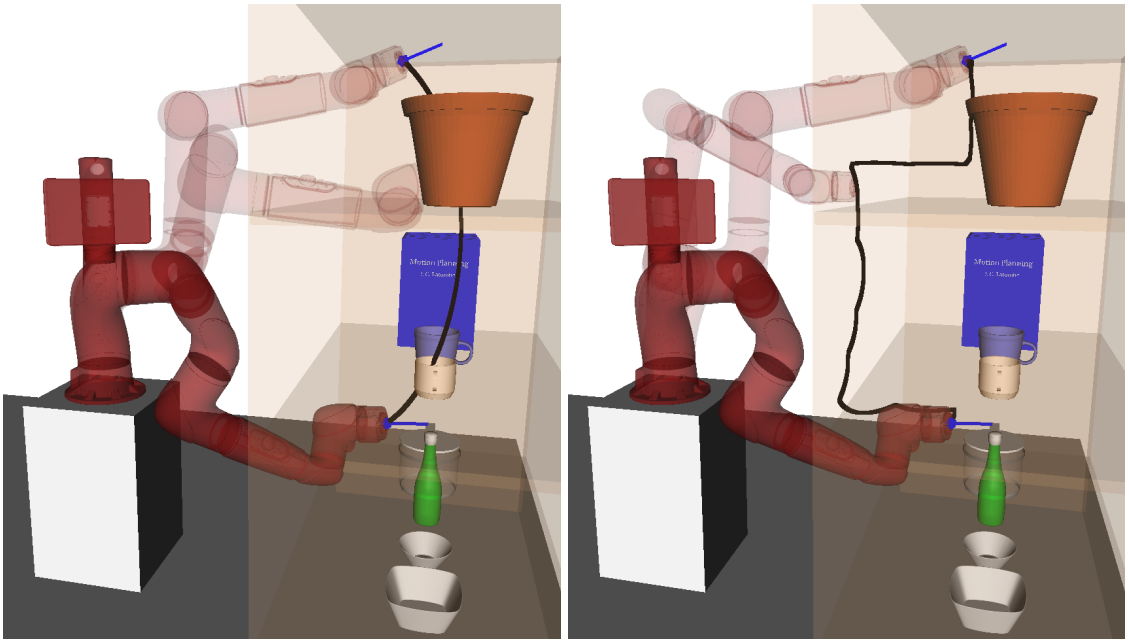
Discussion

This chapter discusses and motivates several aspects of the various component algorithms described throughout the thesis. First, further justification is given for finding multiple HAs in the HAP framework with illustrative examples. Then some insights are provided into why HAP performs better than other methods with some specific examples from the experiments, highlighting why using a coarse estimate for motion costs can be problematic. Then it is discussed why the proposed active perception framework is a fundamental step forward towards solving general robotic perception problems. Finally, the manipulation for perception experiments are discussed and the challenges that lie ahead for solving such problems are analysed.

6.1 Importance of Good Trajectory Initialisation

A key contribution to the high success rate of our HAP approach over naive methods is the utilisation of trajectory initialisations that are informed by an a priori model of the environment. In the case of the bookshelf environment, this is the inflated model of the empty bookshelf.

As illustrated in Figure 6.1a, the naive straight line trajectory initialisation passes through both the bookshelf and objects on the shelves. This is a bad initialisation for a trajectory optimiser as it exhibits many issues that can cause the optimiser to become stuck in

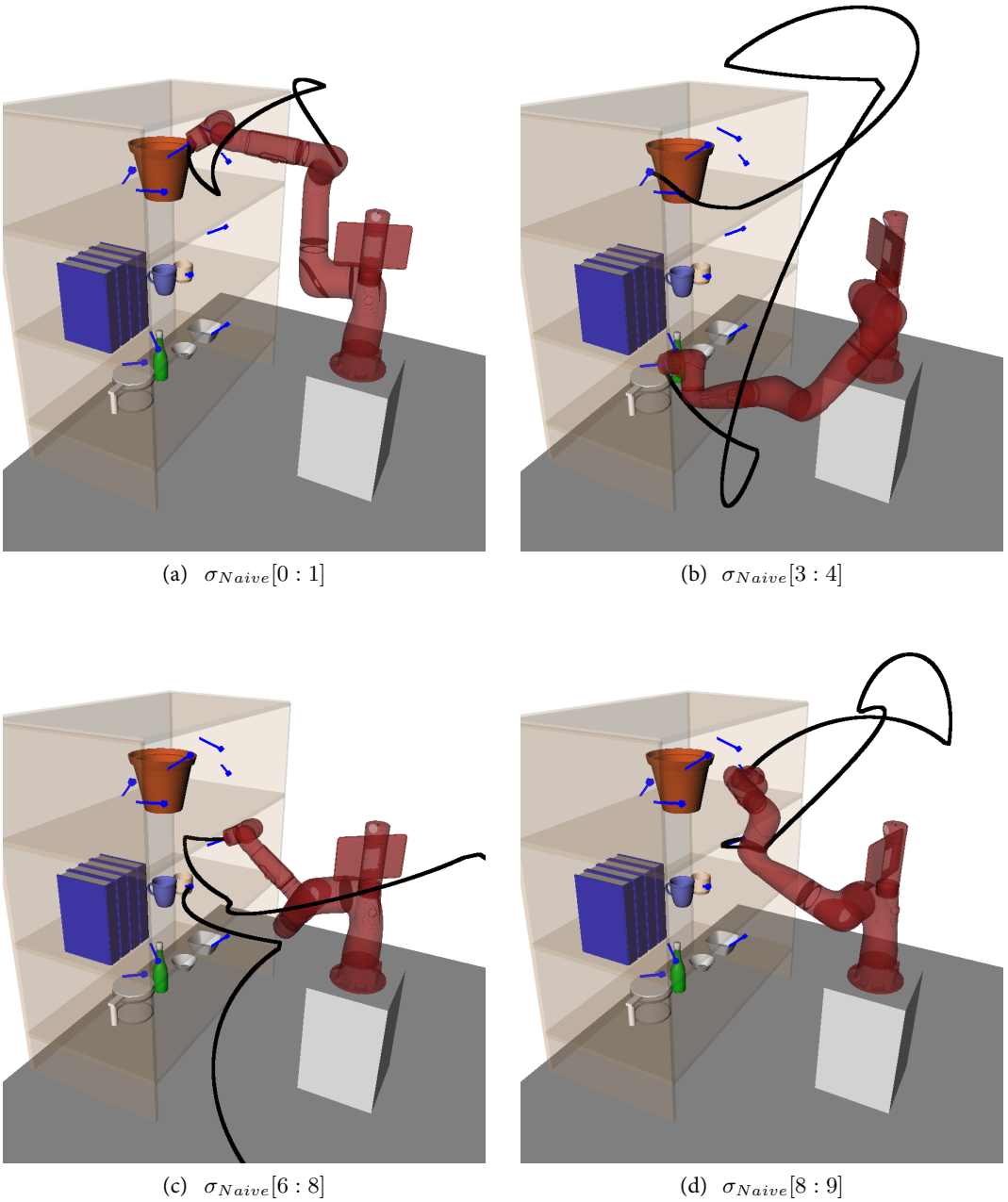


(a) (b)

Figure 6.1: Side view of bookshelf environment showing comparison of (a) naive planner's initial trajectory seed and (b) HAP's. Since HAP has prior knowledge of the environment it seeds with a trajectory that moves away from the shelves before moving into another row.

local minima, such as links being in collision with multiple objects and multiple links having inconsistent gradient directions for pushing them out of obstacles [53]. The shelves themselves are long and thin obstacles which are known to cause issues with trajectory optimisers [54]. As a consequence this particular task failed when running TrajOpt with a naive straight line initialisation.

The initialisation in Figure 6.1b is a HAP library trajectory informed by the a priori environment model. As a consequence, the arm moves away from the shelves before approaching a task in another shelf. This initialisation has a higher chance of succeeding when run through TrajOpt, even though it may initially be in collision with other unseen objects on the bookshelf.



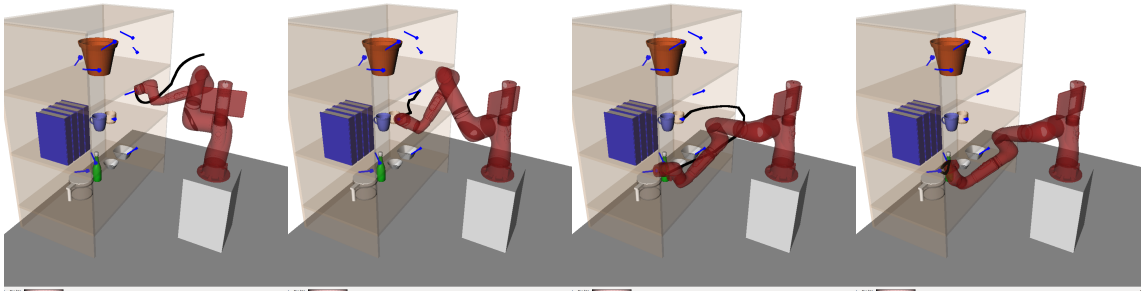
(a) $\sigma_{Naive}[0:1]$ (b) $\sigma_{Naive}[3:4]$

(c) $\sigma_{Naive}[6:8]$ (d) $\sigma_{Naive}[8:9]$

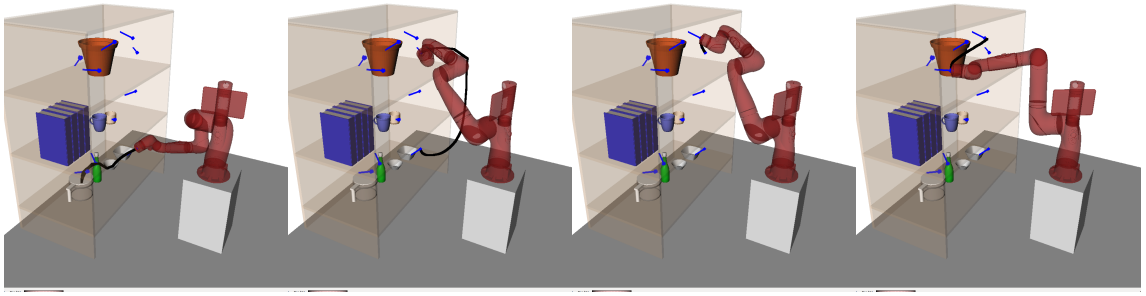
Figure 6.2: Example subset of a Naive-Full trajectory sequence for Sawyer bookshelf experiment. The trajectories for the tasks shown exhibit long and jerky trajectories. (c) skips a task in the sequence since it failed to plan to task 7 (bottom right shelf row).

6.2 Analysing the Behaviour of HAP

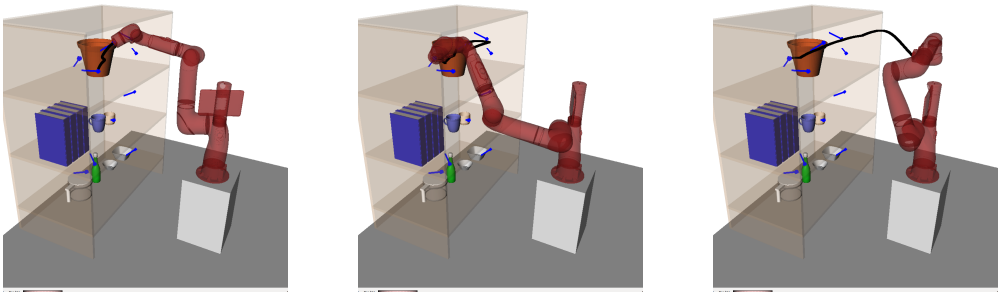
Naive heuristics for planning, such as Euclidean distance in workspace, do not account for the nonlinear motion of high-DOF robotic manipulators. The ϵ-GHAs computed by



(a) $\sigma_{HAP}[0:1]$ (b) $\sigma_{HAP}[1:2]$ (c) $\sigma_{HAP}[2:3]$ (d) $\sigma_{HAP}[3:4]$



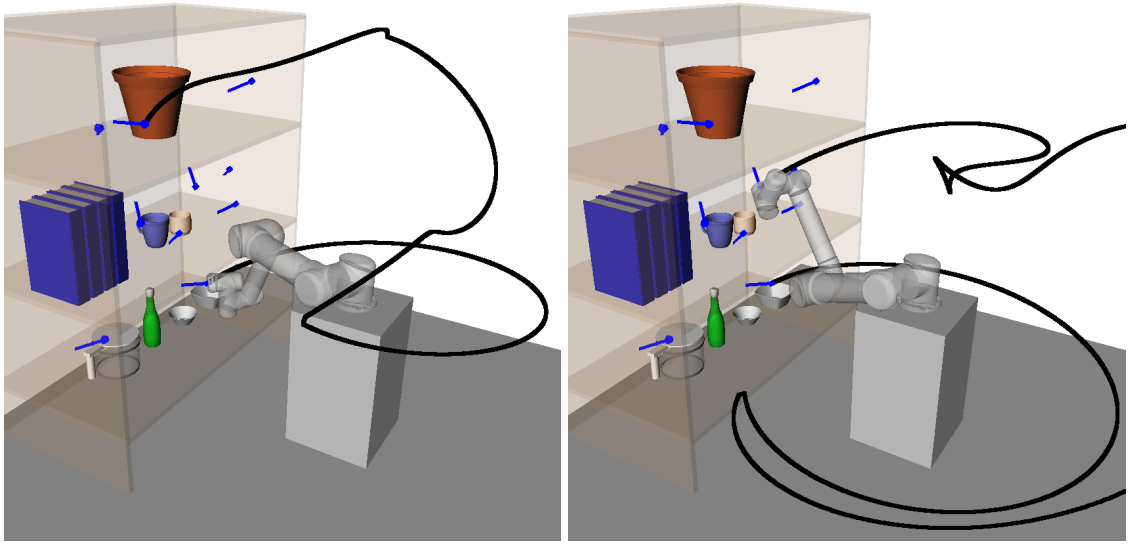
(e) $\sigma_{HAP}[4:5]$ (f) $\sigma_{HAP}[5:6]$ (g) $\sigma_{HAP}[6:7]$ (h) $\sigma_{HAP}[7:8]$



(i) $\sigma_{HAP}[8:9]$ (j) $\sigma_{HAP}[9:10]$ (k) $\sigma_{HAP}[10:0]$

Figure 6.3: Example HAP-Full trajectory sequence for fixed-base model bookshelf experiment. All tasks are successfully planned and trajectories appear to be consistently short and smooth.

HAP account for this nonlinearity by considering prior knowledge such as the kinematic structure and environment obstacles. As a result, HAP achieves improved performance, exemplified in Fig. 6.3 where HAP begins tasks in the middle shelf and sequences tasks according to shelf rows. In contrast, the naive planner executes an inefficient sequence depicted in Fig. 6.2, beginning in the top shelf, continuing tasks in other shelves and returning to the top shelf to complete the sequence.



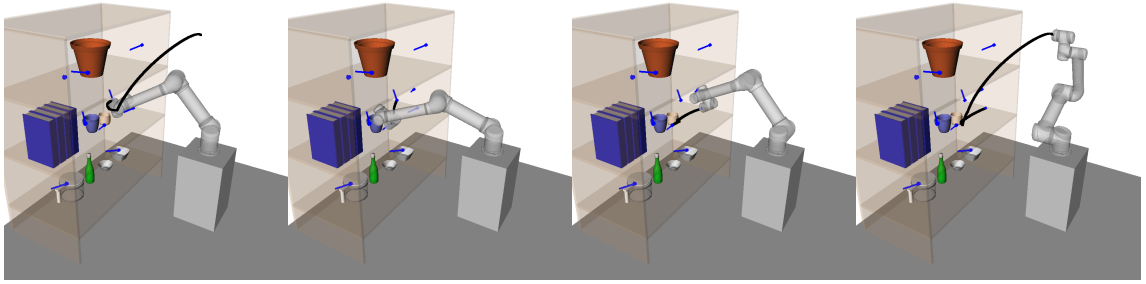
(a) $\sigma_{Naive}[1:3]$ (b) $\sigma_{Naive}[3:5]$

Figure 6.4: Example subset of a Naive-Mobile trajectory sequence for mobile-base model bookshelf experiment. The trajectories for the tasks shown exhibit long jerky trajectories. Both (a) and (b) skip a task in the sequence since they failed to plan to two tasks in the middle and lower shelf rows.

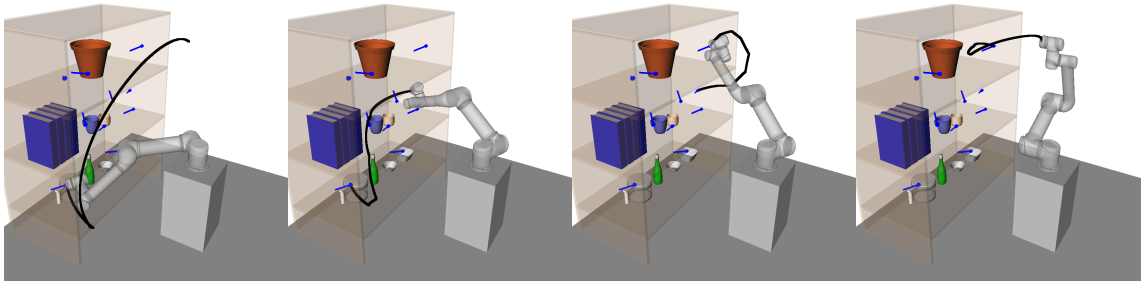
Furthermore, naive trajectories can exhibit excessive jerk as the arm approaches awkward goal configurations along the sequence. In comparison, HAP maintains smooth trajectories over the entire sequence by biasing trajectories to the mapped subspaces. In Fig. 6.2(c), the naive planner failed in one of the tasks in the lower shelf.

The mobile base scenario shown in Figs. 6.4 and 6.5 further demonstrates the benefit of our proposed approach. HAP allocates tasks on the lower shelf to a base pose that allows for more space for the arm to move (Fig. 6.5(j)) while still efficiently grouping tasks in the other rows. Notably, the naive method fails to plan to one such task on the lower shelf.

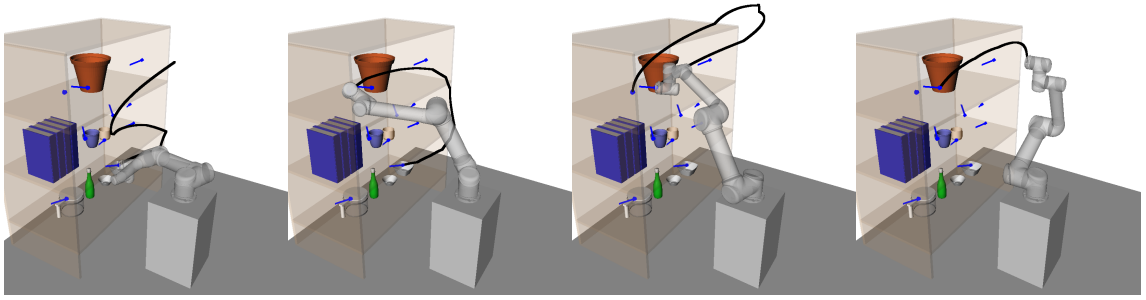
Interestingly, the trajectory in Fig. 6.5(k) required a large rotation about the wrist joint. This is due to the pot object in the top shelf row, which was not known during trajectory library construction. This example indicates that splitting or merging of subspaces may be a direction of future work that could improve the robustness of the framework.



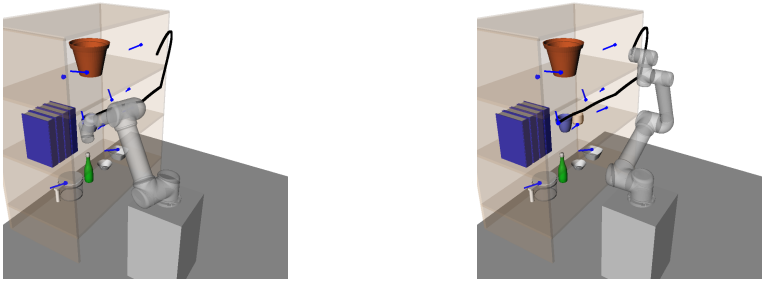
(a) $\sigma_{HAP}[0 : 1], \theta^0$ (b) $\sigma_{HAP}[1 : 2], \theta^0$ (c) $\sigma_{HAP}[2 : 3], \theta^0$ (d) $\sigma_{HAP}[3 : 0], \theta^0$



(e) $\sigma_{HAP}[0 : 4], \theta^1$ (f) $\sigma_{HAP}[4 : 5], \theta^1$ (g) $\sigma_{HAP}[5 : 6], \theta^1$ (h) $\sigma_{HAP}[6 : 0], \theta^1$



(i) $\sigma_{HAP}[0 : 7], \theta^3$ (j) $\sigma_{HAP}[7 : 8], \theta^3$ (k) $\sigma_{HAP}[8 : 9], \theta^3$ (l) $\sigma_{HAP}[9 : 0], \theta^3$



(m) $\sigma_{HAP}[0 : 10], \theta^4$ (n) $\sigma_{HAP}[10 : 0], \theta^4$

Figure 6.5: Example HAP-Mobile trajectory sequence for mobile-base model bookshelf experiment. The assigned ϵ-GHA indexes for each task pair are shown to highlight when subspace switching occurs. All tasks are successfully planned and trajectories appear to be consistently short and smooth.

6.3 Using Easy-To-Find Information to Bootstrap Hard-To-Find Information

A way to interpret the targeted information gathering approach is that it uses easy-to-find information, formalised as ROIs, to bootstrap the process of gathering the hard-to-find information, which is detailed observations of objects from useful viewpoints. Figure 6.6 shows detected ROIs, which are the bright red pixels, overlaid on the apple trellis. Notice that the ROIs include apples, which is desired, and other parts of the image, such as the trellis structure, which is incorrect. This is the trade off of using ROIs; they are very fast to compute, colour thresholding in this example, however they are noisy. As such, they act as an indicator function for data that will be useful for some downstream task such as a more refined yet slower segmentation algorithm, for example the GPIS algorithm we use.

We claim that this approach leads to a fundamental improvement in efficiency. In a sense we are attempting to maximise the collection of potentially useful information while

Figure 6.6: Detected ROIs overlaid on apples trellis.

reducing the amount of unnecessary data, which is a fundamental change to previous reconstruction methods. This is particularly important for non-myopic planning methods such as MCTS which can require many iterations before finding a good solution. This becomes even more burdensome when scaling up to multiple robots. Importantly, we showed that our method enables complex sensor models within these planning methods which is critical to achieving effective performance in complex environments where objects of interest are sparsely located and heavily occluded.

6.4 Manipulation for Perception

The main aim of the manipulation for perception experiment in Section 5.6 was to demonstrate the flexibility of the proposed decentralised coordination framework. Being able to incorporate manipulation into the active perception pipeline opens up the possibility to solve a wider range of problems, especially in extremely cluttered environments where objects are completely occluded from all viewing angles. After carrying out the experiments several challenges were identified and potential approaches for solving these are suggested here.

Identifying parts of the environment to manipulate is challenging. The robot has to determine what can and can't be manipulated which is difficult to infer just from point cloud data. Furthermore, the space of manipulation actions is large and the kinematics and dynamics of the arm need to be considered. Arguably even more difficult is devising an appropriate reward function. One has to consider affordance [30, 200], in other words how manipulating the object affects it and other surrounding parts of the environment. This information then has to be related back to the original perception objective, for example finding ROIs, and quantified in terms of improving the overall objective which is non-trivial. Given the large space of possible scenarios, data driven approaches such as deep learning techniques might be useful for solving such a problem.

Manipulating the environment changes it. This needs to be considered in the environment representation. For example, it was observed during the experiments that while pushing

obstructions out of the way the apples moved around. This was dealt with in the experiment by waiting for a pre-determined amount of time for objects to settle, however, this may be undesirable in a real world application where throughput is important. In some cases objects may not settle back to their original locations and cause the map to become severely corrupted. Hence, a more robust method would be to update the reconstructed map to reflect the movement of the environment.

This problem can be dealt with by dynamically updating the map of the environment [201], however, when moving deformable objects in the environment, such as leaves and branches, it may be more appropriate to use deformable SLAM [202–204] and registration [205, 206] methods because these objects are being moved and deformed as opposed to being removed. However, several of the commonly seen applications in this area deal with environments where the surface is smooth and complete, such as human organs when performing surgery. Furthermore, this is an active research area and many open challenges remain, for example, regarding observability and consistency [207].

In the experiments, one arm with an eye-in-hand sensor is used to manipulate and sense the environment. This is possible if the obstructions are at a far enough distance from the objects being sensed. In order to generalise to any possible obstruction it might be necessary to utilise an additional manipulator with a dedicated gripper for moving the obstructions while the other arm senses and grasps objects. This then becomes a multi-agent problem and our decentralised coordination framework can be utilised for such a scenario.

Furthermore, the demonstrated manipulation actions were non-prehensile; that is, manipulating without grasping [208]. For some obstructions, grasping may be necessary in order to remove them. Additionally, incorporating both grasping and non-prehensile actions could be useful, for example, to simultaneously push obstructions, such as leaves and branches, out of the way in order to grasp objects of interest, such as apples. Ideas from so called "push-grasp" frameworks [209] could be leveraged to achieve this. Furthermore, tactile sensing could be incorporated in order to make grasping objects or moving obstructions out of the way more robust [210].

Despite the above challenges, a promising outlook is that if we can identify parts of the environment to manipulate and compute a reward function for carrying out the manipulation action, then this can seamlessly be integrated into the decentralised coordination framework. The main change would be dynamically updating the action space in the MCTS search, however, this should be a relatively simpler challenge compared to the others outlined above.

Chapter 7

Conclusions and Future Work

Planning for physical systems in the real world that are capable of carrying out valuable work remains a challenge due the large number of components that have to tightly work together in an effective and efficient manner. This is particularly true for complex systems such as teams of robotic manipulators that use their high dexterity to actively sense and manipulate their environments. These systems are desirable as they are general purpose and facilitate performing the most complex tasks that humans carry out in their daily lives. In this thesis we proposed several component algorithms and integrated them into flexible frameworks that are suitable for solving such problems in real-time, reliably and efficiently. Key to achieving this was by making reasonable assumptions about the problem settings and utilising these in the various component algorithms to speed up computation and improve robustness of their behaviour.

For planning the motions of the manipulator and providing constant time cost estimates a novel planning framework called HAP was developed. We showed analytically that HAP provides practical guarantees that greatly increase planning reliability and produce intuitive behaviours. Then a targeted information gathering reward function was developed that effectively balanced between exploration of the scene and targeted observations of regions of interest. These two components were combined into an active perception framework that considers complex motion and sensor models while retaining effective real-time

performance. This was then generalised to a decentralised coordination framework that scales to problems with multiple robots.

These frameworks were evaluated empirically on several different experiments in both simulation and hardware, including robotic pruning, robotic harvesting, and searching for objects in clutter using multiple robot manipulators. It was shown that the proposed methods greatly outperformed naive methods that do not exploit prior knowledge in terms of computation speed, execution efficiency and solution reliability.

In this concluding chapter, a summary of this thesis is provided in Section 7.1 and a summary of the main contributions in Section 7.2. In Section 7.3 areas of future work are discussed for the work presented in this thesis and finally Section 7.4 concludes this thesis with an outlook on the future direction of these general systems in the real world and where the work in this thesis fits in.

7.1 Thesis Summary

The following section summarises each of the major chapters in this thesis.

7.1.1 Hausdorff Approximation Planner (HAP) (Chapter 3)

This thesis first presented HAP, a manipulator motion planning framework inspired by trajectory library methods which is suitable for real-time planning over long task horizons. Importantly, we showed that trajectories through the generated HAs are guaranteed to be of bounded length which is useful for encouraging smooth, predictable behaviour in the online setting.

HAP was evaluated on several experimental scenarios with a six-DOF, with and without a mobile base, and seven-DOF manipulator. The tasks performed by the robot arms included point-to-point motions in a complex bookshelf environment and task-space control over long sequences. Our experiments showed that HAP greatly improved performance over other methods, including faster planning times and higher planning success rates.

7.1.2 Targeted Information Gathering (Chapter 4)

We then presented a framework for targeted information gathering in cluttered, nonuniform scenes where object perception is assisted by indicative ROIs. This included a novel reward function for evaluating candidate viewpoints of a complex sensor model using quality criteria and penalty terms. Furthermore, this information metric effectively balances between exploration of the scene and exploitation of targeted regions of interest which is critical for the complex scenes we are interested in. Furthermore, HAP was integrated for providing accurate motion cost estimates in constant time. The combination of these enabled the use of complex motion and sensor models within a non-myopic active perception framework which we showed requires several hundreds of iterations to perform well.

The framework was validated on an apple harvesting experiment where a robot manipulator with an eye-in-hand RGB-D and novel gripper assembly was tasked to reconstruct and localise fruit on a realistic artificial trellis and then pick them. We showed that the system was able to reliably locate the objects of interest and pick them with little failures within a given time budget.

7.1.3 Decentralised Coordination Framework (Chapter 5)

We presented a decentralised coordination framework which generalised the work in Chapter 4 to efficiently compute an information-maximal set of plans for a team of robots. This framework was evaluated extensively in simulation and our target information metric outperformed other state-of-the-art metrics for viewpoint planning. We additionally carried out experiments in hardware using multiple robotic manipulators with eye-in-hand sensors which were capable of detecting heavily occluded and sparsely located apples.

Furthermore, to demonstrate the flexibility of the framework, preliminary experiments were carried out where a robot arm was required to manipulate obstructions in order to observe objects of interest. The insight was that if these manipulation actions could be automatically generated and a reward function assigned to them then they could be seamlessly appended to the action space of the tree search algorithm.

7.1.4 Discussion (Chapter 6)

In this chapter the various algorithmic and implementation choices throughout the thesis were discussed. This included the need for assigning multiple HAs in particular scenarios and that HAP synergises naturally with a mobile base which allows for more flexible HA assignments.

Additionally, experimental results were discussed and insights into why the proposed algorithms performed well over other compared state-of-the-art were given. In our discussion of HAP we showed that coarse approximations to the motions costs of high dimensional manipulators can cause undesirable, erratic behaviour which is avoided with our method. Further, we discussed why our active perception framework is a fundamental step forward towards solving general robotic perception problems. Lastly, the manipulation for perception experiments were discussed with focus on existing challenges and steps toward solving them.

7.2 Summary of Contributions

We now discuss the principle contributions foreshadowed in Chapter 1.

7.2.1 HAP Framework

This thesis presented a novel manipulator planning framework called HAP that bridges the gaps in existing planning methods. In particular, it overcomes the trade-off between being fast, reliable and efficient which are traditionally conflicting behaviours for general motion planners. It achieves this by leveraging prior knowledge such as environment, kinematic and task structure. Importantly, while other methods aim to achieve similar performance gains by leveraging domain knowledge, ours retains practical guarantees and hence is suitable for applications where predictable and intuitive behaviours are paramount.

Furthermore, HAP is suitable for real world applications where it is common for robots to be performing tasks over long horizons as opposed to single tasks being treated in isolation

of one another. This is particularly important for high-dimensional robotic manipulators that can perform the same task with a diverse range of solutions. The choice of solution is important due to the physical constraints of the system such as joint limits and kinematic singularities. HAP overcomes this by finding HAs which are bijective mappings from task to configuration space, enabling motion planning in the task space which greatly reduces problem complexity and increases robustness over long trajectory sequences.

7.2.2 Targeted Information Gathering VI Metric

In this thesis we presented a novel targeted information metric suitable for reconstructing objects of interest in cluttered environments. The VI metric accounts for occlusions and effectively balances between exploration of the scene and targeted observations of ROIs in the scene. It achieves this by utilising a pinhole camera model to predict what the sensor will see from a particular viewpoint and then computes a VI metric which aims to uncover partially observed ROIs while penalising viewpoints that overlap with already observed ROIs. The proposed approach leads to a fundamental improvement in efficiency in contrast to previous reconstruction methods.

7.2.3 Active Perception Framework

The novel targeted information gathering VI metric and HAP were combined and integrated into an MCTS algorithm to form a real-time, non-myopic active perception framework for solving a wide range of manipulator perception tasks. This framework overcomes the trade off between a complex motion and sensor model which is typically done in order to retain real-time performance. HAP is utilised for querying accurate motion cost estimates of the manipulator in constant time. The proposed VI metric produces an efficient sequence in combination with UCT to maximise information gain given a limited resource budget. The proposed environment model further increases computation efficiency by splitting ROIs and an exploration prior into two separate data structures. This permits a higher resolution ROI sensor model within the non-myopic framework which is crucial when searching for sparse and heavily occluded ROIs.

7.2.4 Decentralised Coordination Framework

We presented a decentralised coordination framework which generalises the proposed active perception framework to multiple robots. This framework leveraged Dec-MCTS with some algorithmic adjustments for increasing computation speed. Importantly, it retains all the desirable properties of the single robot framework and scales linearly to multiple robots. The framework is capable of solving general multiple robot problems, even those that require manipulation of the environment within the active perception pipeline.

7.2.5 Theoretical Analysis

One of the main characteristics of the HAP framework is that it provides desirable practical guarantees which are critical for exhibiting predictable and intuitive behaviours. This is achieved by computing ϵ-GHAs, or HAs for short, which ensure that short paths in task space are short in configuration space. In this thesis we showed that HAP does indeed find mappings with the properties of an ϵ-GHA. Then using this property we showed that geodesics through these HAs in task space correspond approximately to geodesics in configuration space, bounded by some ϵ.

Additionally, a time complexity analysis of the HAP algorithm and the decentralised viewpoint planning problem was given. The former showed that the time complexity of the trajectory library construction algorithm grew proportionally with the cardinality of the largest set of IK solutions for any of the tasks in the discrete task space. The latter showed that the decentralised viewpoint planning problem is intractable for exhaustive methods and that an anytime approach, such as our method, is required to be practically feasible.

7.2.6 Empirical Results

Each of the above frameworks are evaluated extensively in both simulation and hardware under varying scenarios and robotic platforms. The results empirically support the analytical claims, show how the algorithms and frameworks can be applied to various systems and problems, and demonstrate the emergent behaviours and performance.

For HAP the framework was first evaluated extensively in simulation in a complex bookshelf scenario and then in a task-space control scenario. We then evaluated the framework in a robotic pruning scenario which involved visual servoing to several cut points with high precision and performing cuts on branches. These experiments showed that HAP lead to a large improvement in performance when compared to naive approaches that do not leverage domain knowledge. In particular, HAP achieved much higher planning and control success rates and overall better behaved trajectories. This was mainly due to the HAs found by HAP which account implicitly for task constraints and allow for reliable concatenation of trajectories over long task horizons.

To evaluate the active perception framework, a robotic harvesting experiment was performed where a manipulator with an eye-in-hand sensor and novel gripper was tasked to find and then grasp apples on a mock-up trellis. This experiment demonstrated that our perception framework was capable of locating apples reliably given a limited time budget and then HAP was able to efficiently sequence and reliably grasp the apples off the trellis.

Finally our decentralised coordination framework was evaluated extensively in simulation and then in hardware. The simulation experiments were designed to evaluate the targeted information gathering VI metric against other state-of-the-art metrics and variants of our own framework. We showed that for the chosen scenario where objects were sparsely located and heavily occluded that our metric outperformed other metrics by consistently finding all the objects over multiple runs. The framework was then validated in hardware where multiple robot arms with eye-in-hand sensors were tasked to collaboratively find apples on an artificial trellis.

7.3 Future Work

7.3.1 HAP Framework

Apart from the applications explored in this thesis, the HAs could be useful for other problem settings. For example, as discussed earlier HAP could be used for optimal base placement for coverage planning applications [143, 144]. Another interesting application

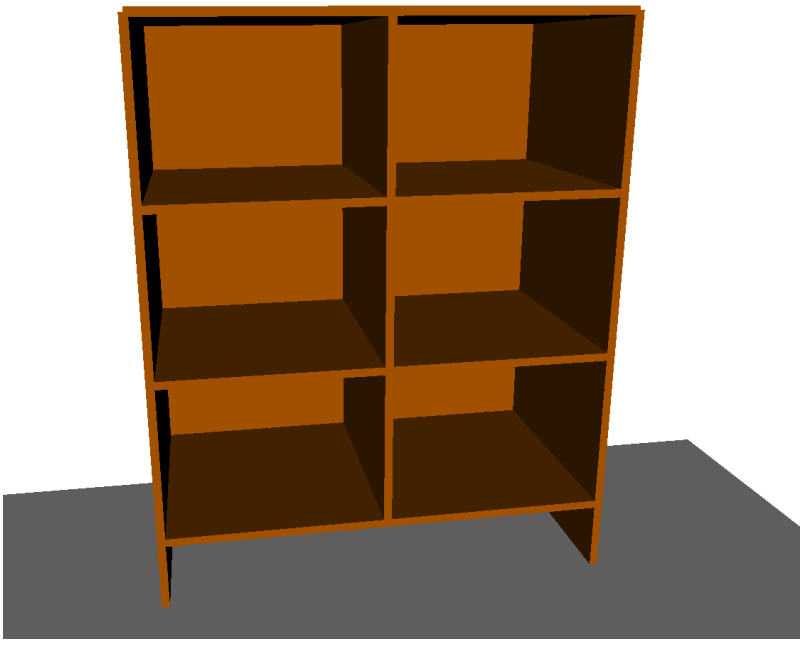


Figure 7.1: Bookshelf environment used in HAP experiments, now with vertical dividing shelf in the centre.

is to provide the HAs found to some local planner or controller, such as RMPflow [89]. Having knowledge of regions of the task space that are approximate isomorphisms is useful for knowing where the arm can move around obstacles comfortably and possibly inform retracting heuristics when moving between tasks in different HAs.

In the current implementation, defining the task space and environment obstacles can be tedious and challenging, particularly for users without expert knowledge. For the former, an interesting research direction is to automatically define the graph representing the task space based on the given environment. For the latter, generating HAs for all possible environments that the robot may encounter can become tedious. Furthermore, new and novel structure may appear in the existing environment. For example in Figure 7.1 the bookshelf environment used in the HAP experiments is modified to have a vertical dividing wall.

An interesting idea is to adapt the original HAs generated for the bookshelf without the divider to the new topology. The benefit in doing so is that previous information can be re-used and the HAs do not need to be computed from scratch again. One approach for

adapting the library is to detect mappings that are in collision due to the new environment, remove them and then compute new shortest paths between nodes within the same HA that were connected by edges to those nodes that were removed. By leveraging dynamic graph search algorithms this could be performed in real-time, allowing for adaptation online.

However, in some cases this may not be possible, for example, when HAs need to be split or merged. Using the bookshelf environment as an example, modifying the number of columns or rows would cause such a scenario to arise. In order to make the trajectory library more generalisable to new environments, online domain adaptation techniques could be explored. In [211], authors used optimal transport (OT) to quickly adapt probabilistic models of laser scans by leveraging a precomputed database of scans. Such a method could be potentially used to adapt the trajectory library to new environments and can handle changes in topology.

Another approach is to learn a distribution of library trajectories conditioned on the environment. A similar approach was used in [212] to predict pedestrian trajectories given a particular environment. Their approach was to train a Mixture Density Network (MDN) to learn a conditional multi-modal distribution of trajectories based on the environment structure. In the context of HAP the modes of the distribution would correspond to the different HAs.

There are several improvements that could be made to the trajectory construction method. For example, the subset of nodes selected for $\hat{T}_{root}$ may have a profound effect on the optimsation routine. Investigating a better approach than random could be a promising avenue for improving performance. Additionally, the library generation is potentially unnecessarily exhaustive and there could be heuristics that reduce computation time by avoiding looking at root nodes or root configurations that won't lead to better HA assignments.

7.3.2 Targeted Information Gathering Framework

There are several avenues for improving and applying the proposed framework to a wider range of problems. For example, the ROIs are manually defined in the problem settings

explored, however, these ROIs could be learnt from data using machine learning techniques. This opens the possibility to generalise to a wider range of problems and even make the perception system more robust to disturbances, such as varying sunlight in outdoor settings.

Another is to consider other planning objectives, such as minimising GPIS segmentation entropy [155], which would require developing approximations for this computationally expensive reward function. This would further improve the reconstruction data by taking into account the downstream segmentation process.

The current pipeline uses a road map of discrete views to plan over and does not consider continuous data collection when moving between viewpoints. This is potentially wasted sensor data that can be collected for free. However, considering this in the perception model increases its complexity and would require an approximate method for predicting what the robot would observe. Another issue is the motion of the manipulator. To achieve continuous observation of the trellis, or regions of interest, the planner has to constrain the orientation of the robot end effector over the whole trajectory. This would require constraint motion planning to be considered, increasing complexity across the whole system since action feasibility would need to be accounted for in the viewpoint planning.

7.3.3 Decentralised Coordination Framework

The demonstrated experiments in this thesis are interesting and demonstrate the capability of the proposed framework, however, there remains a wide range of applications to be explored. For example, other applications include infrastructure inspection [40] and a range of problem settings involving multiple robots with heterogeneous, multi-modal sensors and mobility capabilities [213–215]. Our framework supports these problem settings and testing the framework on a wider range of problems will introduce new and interesting challenges, creating the opportunity to further improve and develop the framework.

Another avenue to explore, as discussed already in Section 6.4, is integrating manipulation in the active perception pipeline where objects are completely occluded and cannot be observed from any viewpoint. Overcoming the challenges discussed and demonstrating a

fully autonomous system would prove that the framework is truly general and lays the foundation to solve almost any problem that humans can.

In the experiments carried out, the manipulators' workspaces were separated into non-overlapping regions. While this greatly simplifies planning, it restricts potential applications where it may be desirable to allow this. For example, in 3D printing applications [216] one may wish to have overlapping workspaces to further increase efficiency. However, this requires collision checking to ensure that robots do not collide which is computationally expensive for high-dimensional manipulators. This is further complicated by real world issues such as sensor uncertainty and execution timing uncertainty which would need to be accounted for in the planning. Incorporating this into the coordination framework is an interesting and challenging problem that would allow for even greater flexibility and performance gains for particular applications.

As for the execution, currently each robot waits for Dec-MCTS to compute a new plan before executing its next action. While this is ideal since it considers the newest sensor information, the robot waits idly for the plan to be computed. Instead, a better execution strategy would be to execute the currently available plan and then switch to the newest plan when it is available. While this increases the complexity of the execution pipeline, and potentially results in less optimal action sequences, the system would execute in a more fluid manner without interruptions.

7.4 Outlook

Interest for robots becoming standard parts of the workforce is increasing in many industries [45]. Their utilisation continues moving away from traditional problems settings, where objects and tasks are fully known beforehand, to more human like scenarios where these dynamically change and even tasks involve collaboratively working with other humans. To handle these scenarios robots must be able to predict and account for human motion and intent.

The frameworks developed in this thesis lay the foundation for solving such problems. For example, in the decentralised coordination framework other humans can be considered

as uncontrolled or under-actuated agents. If one could model the intent [217–222] and potentially even the affordance [223] of interacting with humans for certain tasks then this could be used to predict action sequences, similar to sampling other robot's plans, and consequently to optimise its own plan. An example of this in a real world scenario would be tasking the robot to place an object in a particular pose for a human to work on; for example, to weld onto. Interestingly, the planning objective could consider non-intuitive criteria such as comfort and safety when optimising its plan.

Another interesting aspect to consider is implicit communication. In the context of working with humans the robot may not directly communicate with the robot, hence it has to infer their intent from observation. Such systems are being actively researched [224, 225], however, they only consider scenarios where the other agents are robots as well.

From a motion planning perspective, extending the guarantees exhibited by HAP to dynamic obstacles, such as human motion, would be another important step towards enabling these systems within collaborative environments. Having analytical guarantees about the behaviour of the system is a challenging yet crucial property that should be exhibited by any system working around humans in order to ensure safety and reliability.

Another relevant area of research is biomimics where robots are controlled in a way to imitate biological behaviour, such as natural human motion [226]. This encourages robots to appear more natural and potentially make humans feel more comfortable when working around them or performing tasks collaboratively. Recently there has been a surge of research interest in this area, particularly in the deep learning field [227–230]. Differently, authors in [231] take a more principled approach which looked at imitating humans by learning a manipulability matrix representation which captured "posture preference" when carrying out tasks. This representation is enforced to live in the space of symmetric positive definite (SPD) matrices which, in the context of a manipulator, can be parameterised via a convex cone manifold embedded in $\mathbb{R}^6$. HAP could be used in a similar way to find ϵ-GHA mappings that potentially capture more complex geometric structures than SPD matrices which could then be used to bias trajectories towards more natural human behaviour. There exists a plethora of other metrics used in the literature for encouraging natural human behaviour during motion generation [232–234]. These could be incorporated into

the objective function that HAP uses for optimising the HA mappings by adding them as task-specific costs in a similar fashion to the task-space control applications.

Bibliography

[1] C. Wouter Bac, Eldert J. van Henten, Jochen Hemming, and Yael Edan. Harvesting robots for high-value crops: State-of-the-art review and challenges ahead. *Journal of Field Robotics*, 31(6):888–911, 2014.

[2] Sergey Alatartsev, Sebastian Stellmacher, and Frank Ortmeier. Robotic task sequencing problem: A survey. *Journal of Intelligent & Robotic systems*, 80(2):279–298, 2015.

[3] Matthew Johnson, Brandon Shrewsbury, Sylvain Bertrand, Tingfan Wu, Daniel Duran, Marshall Floyd, Peter Abeles, Douglas Stephen, Nathan Mertins, Alex Lesman, et al. Team ihmc's lessons learned from the darpa robotics challenge trials. *Journal of Field Robotics*, 32(2):192–208, 2015.

[4] Ron Alterovitz, Sven Koenig, and Maxim Likhachev. Robot planning in the real world: Research challenges and opportunities. *Ai Magazine*, 37(2):76–84, 2016.

[5] Nikolaus Correll, Kostas E Bekris, Dmitry Berenson, Oliver Brock, Albert Causo, Kris Hauser, Kei Okada, Alberto Rodriguez, Joseph M Romano, and Peter R Wurman. Analysis and observations from the first amazon picking challenge. *IEEE Transactions on Automation Science and Engineering*, 15(1):172–188, 2016.

[6] Clemens Eppner, Sebastian Höfer, Rico Jonschkowski, Roberto Martín-Martín, Arne Sieverling, Vincent Wall, and Oliver Brock. Lessons from the amazon picking challenge: Four aspects of building robotic systems. In *Proceedings of Robotics: Science and Systems (RSS)*, 2016.

[7] Eric Krotkov, Douglas Hackett, Larry Jackel, Michael Perschbacher, James Pippine, Jesse Strauss, Gill Pratt, and Christopher Orlowski. The darpa robotics challenge finals: Results and perspectives. *Journal of Field Robotics*, 34(2):229–240, 2017.

[8] Christopher G Atkeson, PW Babu Benzun, Nandan Banerjee, Dmitry Berenson, Christoper P Bove, Xiongyi Cui, Mathew DeDonato, Ruixiang Du, Siyuan Feng, Perry Franklin, et al. What happened at the darpa robotics challenge finals. In *The DARPA Robotics Challenge Finals: Humanoid Robots to the Rescue*, pages 667–684. Springer, 2018.

[9] Matthew T Mason. Toward robotic manipulation. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:1–28, 2018.

[10] Ruzena Bajcsy, Yiannis Aloimonos, and John K Tsotsos. Revisiting active perception. *Autonomous Robots*, 42(2):177–196, 2018.

[11] C Liang, KJ Chee, Y Zou, H Zhu, A Causo, S Vidas, T Teng, IM Chen, KH Low, and CC Cheah. Automated robot picking system for e-commerce fulfillment warehouse application. In *Proceedings of International Federation for the Promotion of Mechanism and Machine Science (IFToMM) World Congress*, 2015.

[12] Douglas Morrison, Adam W Tow, Matt Mctaggart, R Smith, Norton Kelly-Boxall, Sean Wade-Mccue, Jordan Erskine, Riccardo Grinover, Alec Gurman, T Hunn, et al. Cartman: The low-cost cartesian manipulator that won the amazon robotics challenge. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 7757–7764, 2018.

[13] Ruud Barth, Jochen Hemming, and Eldert J. Van Henten. Design of an eye-in-hand sensing and servo control framework for harvesting robotics in dense vegetation. *Biosystems Engineering*, 146:71–84, 2016. doi: 10.1016/j.biosystemseng.2015.12.001.

[14] Abhisesh Silwal, Joseph R. Davidson, Manoj Karkee, Changki Mo, Qin Zhang, and Karen Lewis. Design, integration, and field evaluation of a robotic apple harvester. *Journal of Field Robotics*, 2017. doi: 10.1002/rob.21715.

[15] Christopher Lehnert, Andrew English, Christopher McCool, Adam W Tow, and Tristan Perez. Autonomous sweet pepper harvesting for protected cropping systems. *IEEE Robotics and Automation Letters*, 2(2):872–879, 2017.

[16] Pablo Ramon Soria, Fouad Sukkar, Wolfram Martens, B. C. Arrue, and Robert Fitch. Multi-view probabilistic segmentation of pome fruit with a low-cost RGB-D camera. In *Proceedings of ROBOT 2017: Third Iberian Robotics Conference*, pages 320–331. Springer, 2018.

[17] Alexander You, Fouad Sukkar, Robert Fitch, Manoj Karkee, and Joseph R Davidson. An efficient planning and control framework for pruning fruit trees. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 3930–3936, 2020.

[18] Fouad Sukkar, Graeme Best, Chanyeol Yoo, and Robert Fitch. Multi-robot region-of-interest reconstruction with Dec-MCTS. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 9101–9107, 2019.

[19] Geoffrey A. Hollinger and Gaurav S. Sukhatme. Sampling-based robotic information gathering algorithms. *International Journal of Robotics Research*, 33(9):1271–1287, 2014.

[20] Zhan Wei Lim, David Hsu, and Wee Sun Lee. Adaptive informative path planning in metric spaces. *International Journal of Robotics Research*, 35(5):585–598, 2016.

[21] C Wouter Bac, Jochen Hemming, BAJ Tuijl, Ruud Barth, Ehud Wais, and Eldert J Henten. Performance evaluation of a harvesting robot for sweet pepper. *Journal of Field Robotics*, 34(6):1123–1139, 2017.

[22] S. McCammon and G. A. Hollinger. Topological hotspot identification for informative path planning with a marine robot. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[23] Raffaello D'Andrea. Guest editorial: A revolution in the warehouse: A retrospective on kiva systems and the grand challenges ahead. *IEEE Transactions on Automation Science and Engineering*, 9(4):638–639, 2012.

[24] Torgny Brogårdh. Present and future robot control development—an industrial perspective. *Annual Reviews in Control*, 31(1):69–79, 2007.

[25] Cuebong Wong, Erfu Yang, Xiu-Tian Yan, and Dongbing Gu. Adaptive and intelligent navigation of autonomous planetary rovers—a survey. In *Proceedings of NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, pages 237–244, 2017.

[26] Angelos Mallios, Pere Ridao, Marc Carreras, and Emili Hernandez. Navigating and mapping with the sparus auv in a natural and unstructured underwater environment. In *Proceedings of OCEANS'11 MTS/IEEE KONA*, pages 1–7, 2011.

[27] Gautier Dumonteil, Guido Manfredi, Michel Devy, Ambroise Confetti, and Daniel Sidobre. Reactive planning on a collaborative robot for industrial applications. In *Proceedings of International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 2, pages 450–457, 2015.

[28] Siddhartha S Srinivasa, Dave Ferguson, Casey J Helfrich, Dmitry Berenson, Alvaro Collet, Rosen Diankov, Garratt Gallagher, Geoffrey Hollinger, James Kuffner, and Michael Vande Weghe. Herb: a home exploring robotic butler. *Autonomous Robots*, 28(1):5–20, 2010.

[29] Siddhartha S Srinivasa, Dmitry Berenson, Maya Cakmak, Alvaro Collet, Mehmet R Dogar, Anca D Dragan, Ross A Knepper, Tim Niemueller, Kyle Strabala, Mike Vande Weghe, et al. Herb 2.0: Lessons learned from developing a mobile manipulator for the home. *Proceedings of IEEE*, 100(8):2410–2428, 2012.

[30] Mehmet R. Dogar, Kaijen Hsiao†, Matei Ciocarlie†, and Siddhartha S. Srinivasa. Physics-based grasp planning through clutter. In *Proceedings of Robotics: Science and Systems (RSS)*, 2013.

[31] Tom Botterill, Scott Paulin, Richard Green, Samuel Williams, Jessica Lin, Valerie Saxton, Steven Mills, XiaoQi Chen, and Sam Corbett-Davies. A robot system for pruning grape vines. *Journal of Field Robotics*, 2016.

[32] Shigehiko Hayashi, Kenta Shigematsu, Satoshi Yamamoto, Ken Kobayashi, Yasushi Kohno, Junzo Kamata, and Mitsutaka Kurita. Evaluation of a strawberry-harvesting robot in a field test. *Biosystems Engineering*, 105(2):160–171, 2010.

[33] Quoc V Le, Ashutosh Saxena, and Andrew Y Ng. Active perception: Interactive manipulation for improving object detection. *Standford University Journal*, 2008.

[34] Juan Sandino, Fernando Vanegas, Felipe Gonzalez, and Frederic Maire. Autonomous uav navigation for active perception of targets in uncertain and cluttered environments. In *Proceedings of IEEE Aerospace Conference (AeroConf)*, pages 1–12, 2020.

[35] Robert Eidenberger and Josef Scharinger. Active perception and scene modeling by planning with probabilistic 6d object poses. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1036–1043, 2010.

[36] Xinyi Ren, Jianlan Luo, Eugen Solowjow, Juan Aparicio Ojea, Abhishek Gupta, Aviv Tamar, and Pieter Abbeel. Domain randomization for active pose estimation. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 7228–7234, 2019.

[37] C.w. Bac, J. Hemming, and E.j. Van Henten. Robust pixel-based classification of obstacles for robotic harvesting of sweet-pepper. *Computers and Electronics in Agriculture*, 96:148–162, 2013. doi: 10.1016/j.compag.2013.05.004.

[38] Gregory Kahn, Peter Sujan, Sachin Patil, Shaunak Bopardikar, Julian Ryde, Ken Goldberg, and Pieter Abbeel. Active exploration using trajectory optimization for robotic grasping in the presence of occlusions. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 4783–4790, 2015.

[39] Max Schwarz, Anton Milan, Arul Selvam Periyasamy, and Sven Behnke. Rgb-d object detection and semantic segmentation for autonomous manipulation in clutter. *International Journal of Robotics Research*, 37(4-5):437–451, 2018.

[40] Gavin Paul, Stephen Webb, Dikai Liu, and Gamini Dissanayake. Autonomous robot manipulator-based exploration and mapping system for bridge maintenance. *Robotics and Autonomous Systems*, 59(7-8):543–554, 2011.

[41] Stephen Nuske, Sanjiban Choudhury, Sezal Jain, Andrew Chambers, Luke Yoder, Sebastian Scherer, Lyle Chamberlain, Hugh Cover, and Sanjiv Singh. Autonomous exploration and motion planning for an unmanned aerial vehicle navigating rivers. *Journal of Field Robotics*, 32(8):1141–1162, 2015.

[42] Ryan A MacDonald and Stephen L Smith. Active sensing for motion planning in uncertain environments via mutual information policies. *International Journal of Robotics Research*, 38(2-3):146–161, 2019.

[43] Ruzena Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):966–1005, 1988.

[44] Yiannis Aloimonos. *Active perception*. Psychology Press, 2013.

[45] Australian Cobotics Centre. The Australian Cobotics Centre, 2021. URL `https://www.australiancobotics.org`.

[46] Robert J Kosinski. A literature review on reaction time. *Clemson University*, 10(1), 2008.

[47] Fan-Tien Cheng, Jeng-Shi Chen, and Fan-Chu Kung. Study and resolution of singularities for a 7-DOF redundant manipulator. *IEEE Transactions on Industrial Electronics*, 45(3):469–480, 1998. doi: 10.1109/41.679005.

[48] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[49] James J Kuffner and Steven M LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 995–1001, 2000.

[50] Sertac Karaman and Emilio Frazzoli. Incremental sampling-based algorithms for optimal motion planning. In *Proceedings of Robotics: Science and systems (RSS)*, volume 104, 2010.

[51] Kiril Solovey, Lucas Janson, Edward Schmerling, Emilio Frazzoli, and Marco Pavone. Revisiting the asymptotic optimality of rrt. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[52] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa. CHOMP: Covariant hamiltonian optimization for motion planning. *International Journal of Robotics Research*, 32(9-10):1164–1193, 2013.

[53] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *International Journal of Robotics Research*, 33(9):1251–1270, 2014.

[54] Mustafa Mukadam, Jing Dong, Xinyan Yan, Frank Dellaert, and Byron Boots. Continuous-time gaussian process motion planning via probabilistic inference. *International Journal of Robotics Research*, 37(11):1319–1340, 2018.

[55] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[56] Amnart Kanarat. *Motion Planning and robust control for nonholonomic mobile robots under uncertainties*. PhD thesis, Virginia Polytechnic Institute and State University, 2004.

[57] Fouad Sukkar. Fast, reliable and efficient database search motion planner (FREDS-MP) for repetitive manipulator tasks. Master's thesis, University of Technology Sydney, 2018.

[58] Ahmed Hussain Qureshi, Yinglong Miao, Anthony Simeonov, and Michael C Yip. Motion planning networks: Bridging the gap between learning-based and classical motion planners. *IEEE Transactions on Robotics*, 2020.

[59] T. Lai and F. Ramos. PlannerFlows: learning motion samplers with normalising flows. In *Proceedings of IEEE/RSJ International conference on intelligent robots and systems (IROS)*, 2021.

[60] Benjamin Hepp, Debadeepta Dey, Sudipta N Sinha, Ashish Kapoor, Neel Joshi, and Otmar Hilliges. Learn-to-score: Efficient 3D scene exploration by predicting view utility. In *Proceedings of European conference on computer vision (ECCV)*, pages 437–452, 2018.

[61] Dinesh Jayaraman and Kristen Grauman. Learning to look around: Intelligently exploring unseen environments for unknown tasks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1238–1247, 2018.

[62] Yu-Chi Lin and Dmitry Berenson. Using previous experience for humanoid navigation planning. In *Proceedings of IEEE International Conference on Humanoid Robots (Humanoids))*, pages 794–801, 2016.

[63] Lars-Peter Ellekilde and Henrik Gordon Petersen. Motion planning efficient trajectories for industrial bin-picking. *International Journal of Robotics Research*, 32 (9-10):991–1004, 2013.

[64] James Ju Heon Lee, Kris Frey, Robert Fitch, and Salah Sukkarieh. Fast path planning for precision weeding. In *Proceedings of Australasian Conference of Robotics and Automation (ACRA)*, 2014.

[65] Howie M Choset. *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[66] Steven M LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[67] Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 639–646, 2014.

[68] Marc Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1930–1936, 2015.

[69] Sergey Alatartsev, Vera Mersheeva, Marcus Augustine, and Frank Ortmeier. On optimizing a sequence of robotic tasks. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 217–223, 2013.

[70] Andras Kovacs. Task sequencing for remote laser welding in the automotive industry. In *Proceedings of International Conference on Automated Planning and Scheduling (ICAPS)*, 2013.

[71] Ewa Kolakowska, Stephen F Smith, and Morten Kristiansen. Constraint optimization model of a scheduling problem for a robotic arm in automatic systems. *Robotics and Autonomous Systems*, 62(2):267–280, 2014.

[72] Andrs Kovcs. Integrated task sequencing and path planning for robotic remote laser welding. *International Journal of Production Research*, 54(4):1210–1224, 2016.

[73] Christian Wurll and Dominik Henrich. Point-to-point and multi-goal path planning for industrial robots. *Journal of Robotic Systems*, 18(8):445–461, 2001.

[74] Jan Faigl, Vojtech Vonasek, and Libor Preucil. A multi-goal path planning for goal regions in the polygonal domain. In *Proceedings of IEEE European Conference on Mobile Robots (ECMR)*, pages 171–176, 2011.

[75] H. van Hoof, O. Kroemer, and J. Peters. Probabilistic segmentation and targeted exploration of objects in cluttered environments. *IEEE Transactions on Robotics*, 30(5):1198–1209, 2014.

[76] Kanzhi Wu, Ravindra Ranasinghe, and Gamini Dissanayake. Active recognition and pose estimation of household objects in clutter. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 4230–4237, 2015.

[77] T. Patten, M. Zillich, R. Fitch, M. Vincze, and S. Sukkarieh. Viewpoint evaluation for online 3-D active object classification. *IEEE Robotics and Automation Letters*, 1(1):73–81, 2016.

[78] Wen Yu. *PID control with intelligent compensation for exoskeleton robots*. Academic Press, 2018.

[79] Bradley J Nelson and Pradeep K Khosla. Strategies for increasing the tracking region of an eye-in-hand system by singularity and joint limit avoidance. *International Journal of Robotics Research*, 14(3):255–269, 1995.

[80] Jingguo Wang, Yangmin Li, and Xinhua Zhao. Inverse kinematics and control of a 7-dof redundant manipulator based on the closed-loop algorithm. *International Journal of Advanced Robotic Systems*, 7(4):37, 2010.

[81] Graeme Best. Planning algorithms for multi-robot active perception. 2019.

[82] Graeme Best, Oliver Cliff, Timothy Patten, Ramgopal R. Mettu, and Robert Fitch. Dec-MCTS: Decentralized planning for multi-robot active perception. *International Journal of Robotics Research*, 2018. doi:10.1177/0278364918755924.

[83] Graeme Best and Geoffrey A Hollinger. Decentralised self-organising maps for multi-robot information gathering. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4790–4797, 2020.

[84] Noa Agmon, Chien-Liang Fok, Yehuda Emaliah, Peter Stone, Christine Julien, and Sriram Vishwanath. On coordination in practical multi-robot patrol. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 650–656, 2012.

[85] Jeffrey Delmerico, Stefan Isler, Reza Sabzevari, and Davide Scaramuzza. A comparison of volumetric information gain metrics for active 3D object reconstruction. *Autonomous Robots*, 42(2):197–208, 2018.

[86] Ryo Terasawa, Yuka Ariki, Takuya Narihira, Toshimitsu Tsuboi, and Kenichiro Nagasaka. 3D-CNN based heuristic guided task-space planner for faster motion planning. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 9548–9554, 2020.

[87] Alexander Shkolnik and Russ Tedrake. Path planning in 1000+ dimensions using a task-space voronoi bias. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 2061–2067, 2009.

[88] Yu-Chi Lin and Dmitry Berenson. Long-horizon humanoid navigation planning using traversability estimates and previous experience. *Autonomous Robots*, 45(6):937–956, 2021.

[89] Ching-An Cheng, Mustafa Mukadam, Jan Issac, Stan Birchfield, Dieter Fox, Byron Boots, and Nathan Ratliff. Rmpflow: A geometric framework for generation of multitask motion policies. *IEEE Transactions on Automation Science and Engineering*, 2021.

[90] Takayuki Osa. Multimodal trajectory optimization for motion planning. *International Journal of Robotics Research*, 39(8):983–1001, 2020.

[91] William Reid, Robert Fitch, Ali H Gżktoğan, and Salah Sukkarieh. Sampling-based hierarchical motion planning for a reconfigurable wheel-on-leg planetary analogue exploration rover. *Journal of Field Robotics*, 37(5):786–811, 2020.

[92] Mohamed Elbanhawi and Milan Simic. Sampling-based robot motion planning: A review. *IEEE Access*, 2:56–77, 2014.

[93] Jonathan D Gammell and Marlin P Strub. A survey of asymptotically optimal sampling-based motion planning methods. *arXiv preprint arXiv:2009.10484*, 2020.

[94] Michael L Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34(3): 596–615, 1987.

[95] Sven Koenig and Maxim Likhachev. Improved fast replanning for robot navigation in unknown terrain. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 968–975, 2002.

[96] Dave Ferguson, Maxim Likhachev, and Anthony Stentz. A guide to heuristic-based path planning. In *Proceedings of International conference on automated planning and scheduling (ICAPS)*, pages 9–18, 2005.

[97] Dave Ferguson and Anthony Stentz. Field D*: An interpolation-based path planner and replanner. *Robotics Research*, pages 239–253, 2007.

[98] Alex Nash, Kenny Daniel, Sven Koenig, and Ariel Felner. Theta*: Any-angle path planning on grids. In *Proceedings of Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1177–1183, 2007.

[99] Alex Nash, Sven Koenig, and Craig Tovey. Lazy Theta*: Any-angle path planning and path length analysis in 3d. In *Proceedings of International Symposium on Combinatorial Search (SoCS)*, 2010.

[100] Peter Yap, Neil Burch, Robert C. Holte, and Jonathan Schaeffer. Block A*: Database-driven search with applications in any-angle path-planning. In *Proceedings of Association for the Advancement of Artificial Intelligence (AAAI)*, 2011.

[101] Daniel Harabor and Alban Grastien. An optimal any-angle pathfinding algorithm. In *Proceedings of International Conference on Automated Planning and Scheduling (ICAPS)*, 2013.

[102] Thierry Sim'eon, J-P Laumond, and Carole Nissoux. Visibility-based probabilistic roadmaps for motion planning. *Advanced Robotics*, 14(6):477–493, 2000.

[103] Val'erie Boor, Mark H Overmars, and A Frank Van Der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1018–1023, 1999.

[104] Robert Bohlin and Lydia E Kavraki. Path planning using lazy PRM. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 521–528, 2000.

[105] David Hsu. *Randomized single-query motion planning in expansive spaces*. phdthesis, Stanford University USA, 2000.

[106] Gildardo Sanchez and Jean-Claude Latombe. On delaying collision checking in PRM planning: Application to multi-robot coordination. *International Journal of Robotics Research*, 21(1):5–26, 2002.

[107] Kris Hauser and Victor Ng-Thow-Hing. Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 2493–2498, 2010.

[108] Jia Pan, Liangjun Zhang, and Dinesh Manocha. Fast smoothing of motion planning trajectories using b-splines. In *Proceedings of Robotics: Science and Systems (RSS)*, 2011.

[109] Anna Yershova and Steven M LaValle. Improving motion-planning algorithms by efficient nearest-neighbor searching. *IEEE Transactions on Robotics*, 23(1):151–157, 2007.

[110] Erion Plaku, Kostas E Bekris, Brian Y Chen, Andrew M Ladd, and Lydia E Kavraki. Sampling-based roadmap of trees for parallel motion planning. *IEEE Transactions on Robotics*, 21(4):597–608, 2005.

[111] Nancy M Amato and Lucia K Dale. Probabilistic roadmap methods are embarrassingly parallel. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 688–694, 1999.

[112] Joshua Bialkowski, Sertac Karaman, and Emilio Frazzoli. Massively parallelizing the RRT and the RRT. In *Proceedings of IEEE/RSJ International conference on intelligent robots and systems (IROS)*, pages 3513–3518, 2011.

[113] Maxim Likhachev, Dave Ferguson, Geoff Gordon, Anthony Stentz, and Sebastian Thrun. Anytime search in dynamic graphs. *Artificial Intelligence*, 172(14):1613–1643, 2008.

[114] Alex Nash, Sven Koenig, and Maxim Likhachev. Incremental Phi*: Incremental any-angle path planning on grids. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

[115] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846–894, 2011.

[116] Maxim Likhachev and Anthony Stentz. R* search. In *Proceedings of Association for the Advancement of Artificial Intelligence (AAAI)*, page 23, 2008.

[117] Maxim Likhachev, Geoffrey J Gordon, and Sebastian Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 767–774, 2004.

[118] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2997–3004, 2014.

[119] Lucas Janson, Edward Schmerling, Ashley Clark, and Marco Pavone. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *International Journal of Robotics Research*, 34(7):883–921, 2015.

[120] Oktay Arslan and Panagiotis Tsiotras. Use of relaxation methods in sampling-based algorithms for optimal motion planning. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 2421–2428, 2013.

[121] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 3067–3074, 2015.

[122] Marlin P Strub and Jonathan D Gammell. Advanced BIT* (ABIT*): Sampling-based planning with advanced graph-search techniques. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[123] Ryan Luna, Mark Moll, Julia Badger, and Lydia E Kavraki. A scalable motion planner for high-dimensional kinematic systems. *International Journal of Robotics Research*, 39(4):361–388, 2020.

[124] Florian Hauer and Panagiotis Tsiotras. Deformable rapidly-exploring random trees. In *Proceedings of Robotics: Science and Systems (RSS)*, 2017.

[125] Marc Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 1049–1056, 2009.

[126] Siyuan Feng, X Xinjilefu, Christopher G Atkeson, and Joohyung Kim. Optimization based controller design and implementation for the atlas robot in the darpa robotics challenge finals. In *Proceedings of IEEE International Conference on Humanoid Robots (Humanoids)*, pages 1028–1035, 2015.

[127] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 89–96, 2005.

[128] Matt Zucker, Youngbum Jun, Brittany Killen, Tae-Goo Kim, and Paul Oh. Continuous trajectory optimization for autonomous humanoid door opening. In *Proceedings of IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, pages 1–5, 2013.

[129] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. STOMP: Stochastic trajectory optimization for motion planning. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 4569–4574, 2011.

[130] Chonhyon Park, Jia Pan, and Dinesh Manocha. ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments. In *Proceedings of International conference on automated planning and scheduling (ICAPS)*, 2012.

[131] John Schulman, Jonathan Ho, Alex X Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Proceedings of Robotics: Science and Systems (RSS)*, volume 9, pages 1–10, 2013.

[132] Jing Dong, Mustafa Mukadam, Frank Dellaert, and Byron Boots. Motion planning as probabilistic inference using Gaussian processes and factor graphs. In *Proceedings of Robotics: Science and Systems (RSS)*, 2016.

[133] Abhijeet Tallavajhula, Sanjiban Choudhury, Sebastian Scherer, and Alonzo Kelly. List prediction applied to motion planning. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 213–220, 2016.

[134] Jia Pan, Zhuo Chen, and Pieter Abbeel. Predicting initialization effectiveness for trajectory optimization. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 5183–5190, 2014.

[135] Debadeepta Dey, Tian Yu Liu, Martial Hebert, and J Andrew Bagnell. Contextual sequence prediction with application to control library optimization. In *Proceedings of Robotics: Science and Systems (RSS)*, 2013.

[136] S. Natarajan. Learning initial trajectory using sequence-to-sequence approach to warm start an optimization-based motion planner. In *Proceedings of IEEE/RSJ International conference on intelligent robots and systems (IROS)*, 2021.

[137] Eric Huang, Mustafa Mukadam, Zhen Liu, and Byron Boots. Motion planning with graph-based trajectories and gaussian process inference. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 5591–5598, 2017.

[138] Keshav Kolur, Sahit Chintalapudi, Byron Boots, and Mustafa Mukadam. Online motion planning over multiple homotopy classes with gaussian process inference. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2358–2364, 2019.

[139] Kalyan V. Alwala and Mustafa Mukadam. Joint sampling and trajectory optimization over graphs for online motion planning. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[140] Kris Hauser and Jean-Claude Latombe. Integrating task and prm motion planning: Dealing with many infeasible motion planning queries. In *Proceedings of ICAPS09 Workshop on Bridging the Gap between Task and Motion Planning*, 2009.

[141] Caelan Reed Garrett, Tomas Lozano-Perez, and Leslie Pack Kaelbling. FFRob: Leveraging symbolic planning for efficient task and motion planning. *International Journal of Robotics Research*, 37(1):104–136, 2018.

[142] Danny Driess, Jung-Su Ha, and Marc Toussaint. Deep visual reasoning: Learning to predict action sequences for task and motion planning from an initial scene image. In *Proceedings of Robotics: Science and Systems (RSS)*, 2020.

[143] Mahdi Hassan and Dikai Liu. Simultaneous area partitioning and allocation for complete coverage by multiple autonomous industrial robots. *Autonomous Robots*, 41(8):1609–1628, 2017.

[144] F. Paus, P. Kaiser, N. Vahrenkamp, and T. Asfour. A combined approach for robot placement and coverage path planning for mobile manipulation. In *Proceedings of*

IEEE/RSJ International conference on intelligent robots and systems (IROS), pages 6285–6292, 2017.

[145] SY Chen and YF Li. Vision sensor planning for 3-D model acquisition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 35(5):894–904, 2005.

[146] Luca Morreale, Andrea Romanoni, and Matteo Matteucci. Predicting the next best view for 3D mesh refinement. In *Proceedings of International Conference on Intelligent Autonomous Systems (IAS)*, pages 760–772, 2018.

[147] Simon Kriegel, Christian Rink, Tim Bodenmüller, and Michael Suppa. Efficient next-best-scan planning for autonomous 3D surface reconstruction of unknown objects. *Journal of Real-Time Image Processing*, 10(4):611–631, 2015.

[148] Richard Pito. A solution to the next best view problem for automated surface acquisition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21 (10):1016–1030, 1999.

[149] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013.

[150] Berk Calli, Wouter Caarls, Martijn Wisse, and Pieter P Jonker. Active vision via extremum seeking for robots in unstructured environments: Applications in object recognition and manipulation. *IEEE Transactions on Automation Science and Engineering*, 2018.

[151] Rowan Border, Jonathan D Gammell, and Paul Newman. Surface edge explorer (SEE): Planning next best views directly from 3D observations. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[152] J Irving Vasquez-Gomez, L Enrique Sucar, Rafael Murrieta-Cid, and Efrain Lopez-Damian. Volumetric next-best-view planning for 3D object reconstruction with positioning error. *International Journal of Advanced Robotic Systems*, 11(10):159, 2014.

[153] MT Lozano Albalate, Michel Devy, J Miguel, and S Marti. Perception planning for an exploration task of a 3D environment. In *Proceedings of IEEE International Conference on Pattern Recognition (ICPR)*, volume 3, pages 704–707, 2002.

[154] J Irving Vasquez-Gomez, L Enrique Sucar, Rafael Murrieta-Cid, and Juan-Carlos Herrera-Lozada. Tree-based search of the next best view/state for three-dimensional object reconstruction. *International Journal of Advanced Robotic Systems*, 15(1): 1729881418754575, 2018.

[155] W. Martens, Y. Poffet, P. R. Soria, R. Fitch, and S. Sukkarieh. Geometric priors for Gaussian process implicit surfaces. *IEEE Robotics and Automation Letters*, 2(2): 373–380, 2017.

[156] Joni Pajarinen and Ville Kyrki. Decision making under uncertain segmentations. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[157] S. T. Digumarti, J. Nieto, C. Cadena, R. Siegwart, and P. Beardsley. Automatic segmentation of tree structure from point cloud data. *IEEE Robotics and Automation Letters*, 3(4):3043–3050, 2018.

[158] S. Kriegel, M. Brucker, Z.-C. Marton, T. Bodenmuller, and M. Suppa. Combining object modeling and recognition for active scene exploration. In *Proceedings of IEEE/RSJ International conference on intelligent robots and systems (IROS)*, pages 2384–2391, 2013.

[159] R. Monica and J. Aleotti. Surfel-based next best view planning. *IEEE Robotics and Automation Letters*, 3(4):3324–3331, 2018.

[160] Riccardo Monica and Jacopo Aleotti. Contour-based next-best view planning from point cloud segmentation of unknown objects. *Autonomous Robots*, 42(2):443–458, 2018.

[161] Graeme Best, Jan Faigl, and Robert Fitch. Online planning for multi-robot active perception with self-organising maps. *Autonomous Robots*, 42(4):715–738, 2018.

[162] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[163] Timothy Patten, Wolfram Martens, and Robert Fitch. Monte Carlo planning for active object classification. *Autonomous Robots*, 42(2):391–421, 2018.

[164] Andrew J. Smith, Graeme Best, Javier Yu, and Geoffrey A. Hollinger. Real-time distributed non-myopic task selection for heterogeneous robotic teams. *Autonomous Robots*, 2018. doi:10.1007/s10514-018-9811-9.

[165] Akash Arora, P. Michael Furlong, Robert Fitch, Salah Sukkarieh, and Terrence Fong. Multi-modal active perception for information gathering in science missions. *Autonomous Robots*, 2019. doi:10.1007/s10514-019-09836-5.

[166] Micah Corah and Nathan Michael. Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice. *Autonomous Robots*, 43(2):485–501, 2019.

[167] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.

[168] Shayegan Omidshafiei, Ali-Akbar Agha-Mohammadi, Christopher Amato, Shih-Yuan Liu, Jonathan P How, and John Vian. Decentralized control of multi-robot partially observable Markov decision processes using belief space macro-actions. *International Journal of Robotics Research*, 36(2):231–258, 2017.

[169] Charles E Noon and James C Bean. An efficient transformation of the generalized traveling salesman problem. *INFOR: Information Systems and Operational Research*, 31(1):39–44, 1993.

[170] Rolf Adams and Leanne Bischof. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):641–647, 1994.

[171] Qingming Zhan, Yubin Liang, and Yinghui Xiao. Color-based segmentation of point clouds. *Laser scanning*, 38(3):155–161, 2009.

[172] Michael Barbehenn. A note on the complexity of dijkstra's algorithm for graphs with weighted vertices. *IEEE Transactions on Computers*, 47(2):263, 1998.

[173] Dmitri Burago, Iu D Burago, Yuri Burago, Sergei Ivanov, Sergei V Ivanov, and Sergei A Ivanov. *A course in metric geometry*, volume 33. American Mathematical Soc., 2001.

[174] Rethink Robotics. Sawyer robot, 2017. URL `https://www.rethinkrobotics.com/sawyer`.

[175] Universal Robots. The UR5, 2017. URL `https://www.universal-robots.com/products/ur5-robot`.

[176] Google. Solving TSPs with OR-Tools, 2017. URL `https://developers.google.com/optimization/routing/tsp/tsp`.

[177] John Schulman. TrajOpt: trajectory optimization for motion planning. *Robot Learning Lab*, 2013.

[178] Kavraki Lab. The open motion planning library OMPL, 2017. URL `https://ompl.kavrakilab.org/`.

[179] Rosen Diankov and James Kuffner. Open robotics automation virtual environment (OpenRAVE). URL `http://openrave.org`.

[180] Rosen Diankov. Open robotics automation virtual environment (OpenRAVE) repository, 2017. URL `https://github.com/rdiankov/openrave`.

[181] Michael Koval and Chris Dellin. PrPy, 2017. URL `https://github.com/personalrobotics/prpy`.

[182] Michael Koval, Christopher Dellin, and Matthew Klingensmith. OpenRAVE bindings for OMPL, 2017. URL `https://github.com/personalrobotics/or_ompl`.

[183] Michael Koval and Pras Velagapudi. OpenRAVE prpy bindings for TrajOpt, 2017. URL `https://github.com/personalrobotics/or_trajopt`.

[184] Michael Koval and Pras Velagapudi. OpenRAVE plugin for URDF, 2017. URL `https://github.com/personalrobotics/or_urdf`.

[185] Jia Pan, Sachin Chitta, and Dinesh Manocha. FCL: A general purpose library for collision and proximity queries. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 3859–3866, 2012.

[186] Jose Luis Rivero. FCL - The Flexible Collision Library, 2017. URL `https://github.com/flexible-collision-library/fcl`.

[187] Rethink Robotics. Sawyer robot, 2017. URL `https://github.com/RethinkRobotics/sawyer_robot`.

[188] ROS Industrial. ROS-Industrial Universal Robot meta-package, 2017. URL `https://github.com/ros-industrial/universal_robot`.

[189] William J Wilson, CC Williams Hulls, and Graham S Bell. Relative end-effector control using cartesian position based visual servoing. *IEEE Transactions on Robotics and Automation*, 12(5):684–696, 1996.

[190] Tsuneo Yoshikawa. Manipulability of robotic mechanisms. *International Journal of Robotics Research*, 4(2):3–9, 1985.

[191] Charles A Klein and Bruce E Blaho. Dexterity measures for the design and control of kinematically redundant manipulators. *International Journal of Robotics Research*, 6(2):72–83, 1987.

[192] Donald Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129–147, 1982.

[193] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[194] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Proceedings of European Conference on Machine Learning (ECML)*, pages 282–293, 2006.

[195] Michael Gienger, Herbert Janssen, and Christian Goerick. Task-oriented whole body motion for humanoid robots. In *Proceedings of IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, pages 238–244, 2005.

[196] Hisashi Sugiura, Michael Gienger, Herbert Janssen, and Christian Goerick. Real-time self collision avoidance for humanoids by means of nullspace criteria and task intervals. In *Proceedings of IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, pages 575–580, 2006.

[197] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[198] Stephen Nuske, Kyle Wilshusen, Supreeth Achar, Luke Yoder, and Sanjiv Singh. Automated visual yield estimation in vineyards. *Journal of Field Robotics*, 31(5): 837–860, 2014.

[199] Open Robotics. ROS - robot operating system, 2017. URL `https://ros.org`.

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

[201] Mathieu Labbe and Frančois Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446, 2019.

[202] Jingwei Song, Jun Wang, Liang Zhao, Shoudong Huang, and Gamini Dissanayake. Mis-slam: Real-time large-scale dense deformable slam system in minimal invasive surgery based on heterogeneous computing. *IEEE Robotics and Automation Letters*, 3(4):4068–4075, 2018.

[203] Juan J Gomez-Rodriguez, Jose Lamarca, Javier Morlana, Juan D Tardos, and Jose MM Montiel. Sd-defslam: Semi-direct monocular slam for deformable and intracorporeal scenes. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 5170–5177, 2021.

[204] Shuai Zhang, Liang Zhao, Shoudong Huang, Ruibin Ma, Boni Hu, and Qi Hao. 3d reconstruction of deformable colon structures based on preoperative model and deep neural network. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 1875–1881, 2021.

[205] Shaifali Parashar, Daniel Pizarro, and Adrien Bartoli. Isometric non-rigid shape-from-motion with riemannian geometry solved in linear time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(10):2442–2454, 2017.

[206] Shaifali Parashar, Daniel Pizarro, and Adrien Bartoli. Local deformable 3d reconstruction with cartan's connections. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(12):3011–3026, 2019.

[207] Shoudong Huang, Yongbo Chen, Liang Zhao, Yanhao Zhang, and Mengya Xu. Some research questions for slam in deformable environments. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7653–7660.

[208] Mehmet R Dogar and Siddhartha S Srinivasa. A planning framework for non-prehensile manipulation under clutter and uncertainty. *Autonomous Robots*, 33(3): 217–236, 2012.

[209] Mehmet Dogar and Siddhartha Srinivasa. A framework for push-grasping in clutter. In *Proceedings of Robotics: Science and systems (RSS)*, volume 1, 2011.

[210] Michael C Koval, Nancy S Pollard, and Siddhartha S Srinivasa. Pose estimation for planar contact manipulation with manifold particle filters. *International Journal of Robotics Research*, 34(7):922–945, 2015.

[211] Anthony Tompkins, Ransalu Senanayake, and Fabio Ramos. Online domain adaptation for occupancy mapping. In *Proceedings of Robotics: Science and Systems (RSS)*, 2020.

[212] W. Zhi, T. Lai, L. Ott, and F. Ramos. Trajectory generation in new environments from past experiences. In *Proceedings of IEEE/RSJ International conference on intelligent robots and systems (IROS)*, 2021.

[213] Tim Bailey, Mitch Bryson, Hua Mu, John Vial, Lachlan McCalman, and Hugh Durrant-Whyte. Decentralised cooperative localisation for heterogeneous teams of mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 2859–2865, 2011.

[214] Teresa A Vidal-Calleja, Cyrille Berger, Joan Solà, and Simon Lacroix. Large scale multiple robot visual mapping with heterogeneous landmarks in semi-structured terrain. *Robotics and Autonomous Systems*, 59(9):654–674, 2011.

[215] Yara Rizk, Mariette Awad, and Edward W Tunstel. Cooperative heterogeneous multi-robot systems: A survey. *ACM Computing Surveys*, 52(2):1–31, 2019.

[216] Saivipulteja Elagandula, Laxmi Poudel, Zhenghui Sha, and Wenchao Zhou. Multi-robot path planning for cooperative 3d printing. In *Proceedings of ASME International Manufacturing Science and Engineering Conference (MSEC)*, volume 84256, page V001T01A034, 2020.

[217] Julieta Martinez, Michael J Black, and Javier Romero. On human motion prediction using recurrent neural networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2891–2900, 2017.

[218] Judith Butepage, Michael J Black, Danica Kragic, and Hedvig Kjellstrom. Deep representation learning for human motion prediction and classification. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6158–6166, 2017.

[219] Liang-Yan Gui, Yu-Xiong Wang, Xiaodan Liang, and José MF Moura. Adversarial geometry-aware human motion prediction. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 786–803, 2018.

[220] Alexander Virgona, Alen Alempijevic, and Teresa Vidal-Calleja. Socially constrained tracking in crowded environments using shoulder pose estimates. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 4555–4562, 2018.

[221] Stefan H Kiss, KY To, Chanyeol Yoo, Robert Fitch, and Alen Alempijevic. Minimally invasive social navigation. In *Proceedings of Australasian Conference of Robotics and Automation (ACRA)*, 2019.

[222] Stefan H Kiss, Kavindie Katuwandeniya, Alen Alempijevic, and Teresa Vidal-Calleja. Probabilistic dynamic crowd prediction for social navigation. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 9269–9275, 2021.

[223] Tianmin Shu, Michael S Ryoo, and Song-Chun Zhu. Learning social affordance for human-robot interaction. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.

[224] Benjamin Riviere, Wolfgang Hoenig, Matthew Anderson, and Soon-Jo Chung. Neural tree expansion for multi-robot planning in non-cooperative environments. *IEEE Robotics and Automation Letters*, 6(4):6868–6875, 2021.

[225] Hai Zhu, Francisco Martinez Claramunt, Bruno Brito, and Javier Alonso-Mora. Learning interaction-aware trajectory predictions for decentralized multi-robot motion planning in dynamic environments. *IEEE Robotics and Automation Letters*, 6 (2):2256–2263, 2021.

[226] Jiankun Wang, Weinan Chen, Xiao Xiao, Yangxin Xu, Chenming Li, Xiao Jia, and Max Q-H Meng. A survey of the development of biomimetic intelligence and robotics. *Biomimetic Intelligence and Robotics*, 1:100001, 2021.

[227] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in Neural Information Processing Systems*, 29:4565–4573, 2016.

[228] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Proceedings of Conference on Robot Learning (CoRL)*, pages 357–368, 2017.

[229] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 4693–4700, 2018.

[230] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics*, 37(4):1–14, 2018.

[231] Noemie Jaquier, Leonel Rozo, Darwin G Caldwell, and Sylvain Calinon. Geometry-aware manipulability learning, tracking, and transfer. *International Journal of Robotics Research*, 40(2-3):624–650, 2021.

[232] Minas Liarokapis, Charalampos P Bechlioulis, Panagiotis K Artemiadis, and Kostas J Kyriakopoulos. Deriving humanlike arm hand system poses. *Journal of Mechanisms and Robotics*, 9(1):011012, 2017.

[233] Gianpaolo Gulletta, Wolfram Erlhagen, and Estela Bicho. Human-like arm motion generation: A review. *Robotics*, 9(4):102, 2020.

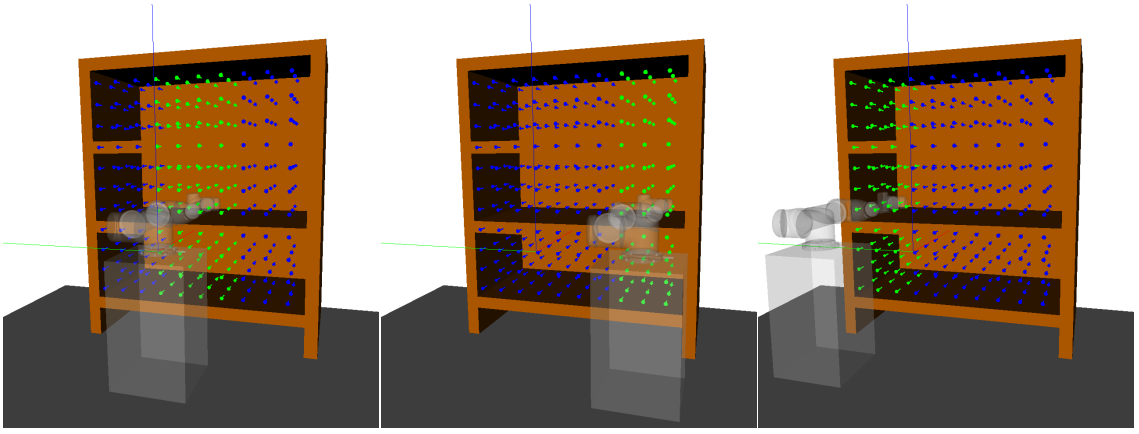
[234] Carl Gaebert, Sascha Kaden, and Ulrike Thomas. Generation of human-like arm motions using sampling-based motion planning. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2534–2541, 2021.

[235] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

Appendices

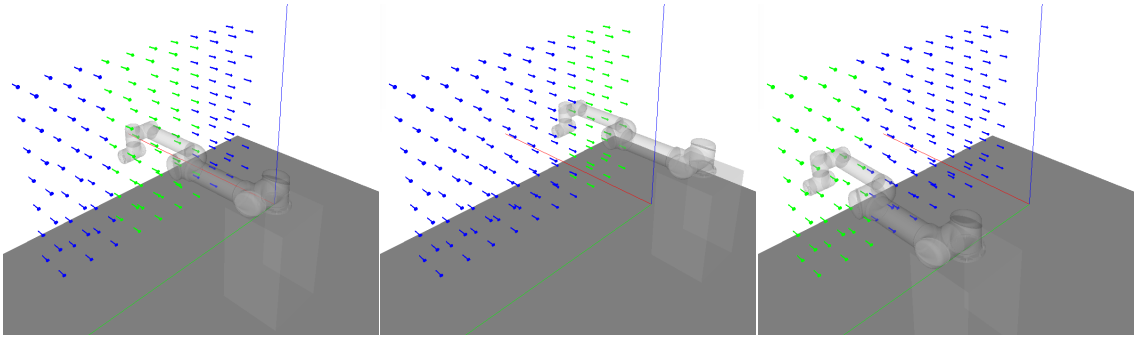
A Experiment Supporting Figures

A.1 Naive Task Region Base Allocations for HAP Experiments



(a) (b) (c)

Figure A.1: Subspaces and base positions (manually defined) for Naive-Mobile in single- and batch-query experiments. The task space is divided into three regions with the intent of reducing the reach distance from each of the base positions.



(a) (b) (c)

Figure A.2: Subspaces and base positions (manually defined) for the Naive-Mobile method in task-space control experiments. Design intention as in previous figure.

A.2 Apple Trellis Configurations for Harvesting Experiments

Figure A.3: Trellis configurations used for apple harvesting experiments. Ordering (row-major) corresponds with experimental result numbering.

B Supplementary Algorithmic and Implementation Details

B.1 Null-Space Control for Avoiding Joint Limits

When performing task-space control, we may wish to exploit the arm's redundancy to improve some constraint based criteria such as joint limit avoidance. This is achieved in (4.10) where a repulsive term, $\dot{q}_0$, was added that is projected into the null-space of the Jacobian. A common choice is [80]:

$$\dot{q}_0 = k_0 \left(\frac{\partial U(q)}{\partial q} \right)^T \tag{B.1}$$

where $U(q)$ is a function of the joint variables to locally maximize. This function can be used for other purposes such as avoiding obstacles or for increasing manipulability during the motion to avoid singularities. In this work we exploit the redundancy for keeping the joints away from the joint limits.

For this purpose, we use the following repulsive potential functions for each joint j:

$$U_j^{upper}(q) = \begin{cases} \frac{1}{2} k_{r_j} \left(\frac{1}{d_j^{upper}} - \frac{1}{Q_j^{upper}} \right)^2 & \text{if } d_j^{upper} \leq Q_j^{upper} \\ 0 & \text{otherwise} \end{cases} \tag{B.2}$$

$$U_j^{lower}(q) = \begin{cases} \frac{1}{2} k_{r_j} \left(\frac{1}{d_j^{lower}} - \frac{1}{Q_j^{lower}} \right)^2 & \text{if } d_j^{lower} \leq Q_j^{lower} \\ 0 & \text{otherwise} \end{cases} \tag{B.3}$$

where $k_{r_j} \in \mathbb{R}$ is a positive gain, d_j^{upper}, d_j^{lower} are the distances of the joint position from, respectively, the upper and the lower joint limit and $Q_j^{upper}, Q_j^{lower} \in \mathbb{R}$ are positive thresholds used to limit the repulsive effect in the neighbourhood of the joint limits. The global repulsive function is $U_{rep}(q) = [U_1^{upper} + U_1^{lower}, \ldots, U_n^{upper} + U_n^{lower}]^T$. It is possible to move away from joint limits by setting:

$$\dot{q}_0 = -\frac{\partial U_{rep}(q)}{\partial q} \tag{B.4}$$

B.2 Upper Confidence Bound 1 Applied to Trees (UCT)

The Monte Carlo Tree Search (MCTS) variant used in the active perception framework in Section 4.5 is called Upper Confidence Bound 1 applied to trees (UCT) [194]. The UCT algorithm performs an asymmetric expansion of a search tree using a best-first policy that generalises the UCB1 policy for multi-armed bandit (MAB) problems [235].

In the UCT formulation, node selection is performed by choosing the node that maximises the following expression:

$$F_i + w\sqrt{\frac{\ln N_i}{n_i}}, \tag{B.5}$$

where F_i is the empirical average reward after iteration i, w is a tunable exploration parameter, n_i is the number of simulations for the node being considered and N_i is the number of simulations for the parent of the node being considered. The first component of the above formula is the exploitation term which is high for actions that result in high reward. The second term is the exploration term which is high for actions with fewer simulations.

B.3 Discounted UCT (D-UCT)

To adapt the active perception framework in Section 4.5 to multiple robots, we utilise Dec-MCTS [82] which is a decentralised version of MCTS. Dec-MCTS proposes an extension to UCT, called Discounted UCT (D-UCT) for selecting child nodes suitable for the modified problem setting. That is, the joint information gain objective in Problem 5.1 is not only a function of an individual robot's action sequence but additionally the unknown action sequence of all other robots.

In a decentralised setting this means that the tree search has to allow for changing reward distributions as other robot's plan distributions are received. The main difference UCT and D-UCT is that a discount factor is added to the formulation in (B.5) which favours more recent rollouts. D-UCT recovers similar analytical results to the UCT algorithm for

this generalised multi-robot case. More details can be found in the original Dec-MCTS paper [82].