3D Adaptive Coverage Path Planning for Autonomous Submarine Tank Inspection Robots

by

Rowan Pivetta

BEng (Robotics)(Hons)

Thesis

Submitted to Flinders University

for the degree of

Doctor of Philosophy

College of Science and Engineering

November 2020

Abstract

The manual inspection of the confined spaces within the Australian Collins Class submarine tanks for evidence of corrosion, paint delamination and potential defects, is a hazardous, time consuming, and expensive process. Automating submarine tank inspections will eliminate the prolonged exposure of human inspectors to the hazardous confined spaces of the submarine. The robotic platform suitable for this task is a six-legged robot that possesses electromagnetic adhesion giving it the ability to climb freely throughout the complex steel tank structures of a submarine.

Coverage path planning algorithms can be used to generate inspection plans, however, in complex environments, this a challenging problem. Coverage planning algorithms can be generated offline prior to inspection, however, these plans are unable to adapt to changes in the environment. An *adaptive coverage planner* that enables the robot to navigate around detected obstacles whilst providing sensory coverage of the newly detected features is preferable. An *adaptive coverage planner* is developed in this thesis to fulfil this requirement and enable an autonomous platform to perform a comprehensive inspection of submarine tanks.

This thesis extends the capability of an existing *offline sampling-based coverage planner*, known for generating discrete coverage plans in complex environments, with online path replanning strategies to perform adaptively during execution. Two replanning strategies were explored, a *full replan* and *plan repair*, neither of which have previously been applied to adaptively update a current inspection plan to changing conditions using the *offline coverage planner*.

An examination of the *offline sampling-based coverage planner* within the representative submarine tank scenario was used to determine its effectiveness for online performance. Key

results showed that the *offline coverage planner* was susceptible to significant variable planning times for large complex planning problems and was sensitive to minor variations in the environment. New methods derived from the analysis were developed to resolve the variability of planning times and sensitivity to the environment, consequently led to the reduction of planning times from 6 hours to under 10 minutes. The results of these improvement indicated that a *plan repair strategy* was best to adapt the *offline coverage planner* to the online domain.

Experiments comparing the two replanning strategies revealed that an *adaptive sampling-based coverage planner* using a *plan repair strategy* was the faster approach compared to a *full replan strategy*. The *plan repair strategy* demonstrated its capability of replanning up to 92% of the existing tour without significant degradation, completing the replanning problems for the representative submarine tank environment 396 times faster, going from 46 hours to 7.5 minutes.

Declaration

I certify that this thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief, does not contain any material previously published or written by another person except where due reference is made in the text.

Rowan Pivetta

November 10, 2020

Acknowledgements

When you do something for so long you encounter many people along the way that contribute in so many different ways and I would like to take the time to thank them all because it does not matter how big or small the contribution I have appreciated it all. You have all given me invaluable insight and input into my work and each one of you have made a significant impact on me personally and professionally.

Firstly, I would like to thank my principle supervisor Professor Karl Sammut for having me as a PhD candidate and recommending me to undertake the Submarine Tank Inspection project as a part of my thesis.

I would like to thank Professor Jonathan Binns and Dr. Elizabeth Vagg from the Research Training Centre for Naval Design and Manufacturing for their co-ordination of the industry lead PhD program.

I would like to acknowledge industry partners ASC Pty Ltd, research collaborators from the University of Wollongong, Andrew Short and Associate Professor Stephen van Duin, Flinders University and the support I received as a recipient of the Australian Government Research Training Program Scholarship.

My fellow and former colleagues of the Centre of Maritime Engineering at Flinders University, Dr. Candice Francis, Bradley Donnelly, Dr. Amir Yazdani. It was a pleasure to be a part of the group and I cherish the support you gave to me at various stages throughout my candidature.

To the academic staff that were not even associated with my thesis but never passed on the opportunity to check up on my progress, Associate Professor Fangpo He, Dr. David Hobbs, Associate Professor Kenneth Pope and Dr. Vlatka Zivotic-Kuklj. I thank you for your

guidance and wisdom at various stages of my candidature. Your support had a profound impact on my progress and professional development as a young researcher.

Thank you Dr Lisa Schultz for your time and advice on statistical methods.

To Darren Lohmeyer and Dr Maria John, I thank-you for your gracious understanding of the pressures of bringing together a thesis by allowing me to take the time from my internship to complete it. I would not have been in a position without it to produce a thesis that I have become proud of.

I would like to thank Colin Wardle, Dr. Mikael Johansson, Trevor Reschke, Jack Atkinson and Dr. Margaret Law. You made me quite welcome within your team and I thank you for providing me with the opportunity to work at ASC. A special acknowledgement needs to go to Colin Wardle. Coming into my PhD with minimal software experience, your time and experience provided me with the insights I needed to develop my ideas and algorithms.

To my fellow Level Five friends, Dr. Dermot O'Rourke, Dr. Dhara Amin, Dr. Albert Ruiz-Vargas, Dr. Bryant Roberts, Dr. Mark Gardner, Dr. Laura Gell, Dr. Maged Awadalla, Dr. Isabelle Van Merilo, Andrea Menichetti, Francesca Bucci, Sophie Rapagna, Robbie Trott, Michael Russo and Marco Branni, I thank-you for your companionship throughout my candidature. To such an incredible bunch of people, for whom I have great admiration, it has been a pleasure to be a part of each and every one of your PhD journeys as well.

To Lee Ying Wu, I appreciate the time and support you provided me at the early stages of my PhD. You helped with my software development and gave me a great foundation to take my ideas further. I would like to formally acknowledge the improvement that Lee provided me that reorganised my nearest neighbours search tree to search by primitive incentre rather than primitive vertices (Section 4.2.1). Without this, I would still be waiting for my results from first year.

To James Armitage, I could not have asked for a better research assistant and friend. I appreciate all the effort you poured into developing the backend software that formed the foundation of the *hybrid-heuristic*. You completed this work with such rigor and dedication, and I am extremely proud of what we have achieved. Formally, I would like to credit James with the *Rule 5 amendment* that enabled viewing locations to be connected directly to the skeleton during the thinning process (Section 6.5.1). This improvement enabled my idea of

skeletonisation in complex spaces to work incredibly well for large complex planning problems and for that I am grateful.

To Dr. Louise Pelle, I thank you for consistently checking up on me over the last months of my candidature, even up to the final hours. Our goal setting routine each day gave me the focus I needed to get through each and every challenging aspect of completing a thesis.

To Dr. Dermot O'Rourke, you set up those morning writing sessions to get me writing my thesis, and it worked, obviously. These sessions helped me overcome my biggest hurdle. I appreciate the time you put aside, gave me the motivation to make all this happen and I will cherish these times, we had a lot of fun.

To my advisory team Dr. Jonathan Wheare, Dr. Andrew Lammas and Richard Bowyer. They say good things come in threes and I do not believe I could have asked for a better team. What you have given me is more than an acknowledgement or a simple thank you can say. You never once declined my request for help, and your outstanding commitment of getting me through my thesis has been absolutely invaluable. To Jonathan, you always looked out for me, always asked how I was going and was always up for a good chinwag. You have been a fantastic sounding board and collaborator, always willing to listen to my ideas. Andrew, as my assistant supervisor and post-doc on the Submarine Tank Inspection project, you have been a pillar of support throughout my entire candidature. To a brilliant minded and talented individual, it has been a pleasure working alongside you. Richard, I thank you for your unwavering support. You kept me grounded, and focused. You always knew what to say and how to help. You helped me grow as a person both professionally and personally.

Finally, to my family, Kathy, Lui, Caidan, Kazia, Kyra and Jalina. I could not have finished this thesis without the continual love and support you gave to me throughout my PhD and all my studies. I would not be here without you. Over my long course of study, you have always provided everything I needed to get through and for that I am forever grateful. As such, I dedicate this thesis to you.

Table of Contents

List of Figures **xiv**

List of Tables **xxvi**

Glossary **xxix**

Nomenclature **xxxii**

Publications **xxxv**

Chapter 1 Towards Automated Submarine Tank Inspection **1**

1.1 Motivation 1

1.2 Robotic Platforms for Confined Space Inspection 4

1.2.1 Multi-legged Platforms 6

1.2.2 Concept Demonstrator 8

1.3 Processes for an Autonomous Inspection Platform 9

1.3.1 Mapping System 10

1.3.2 Inspection Planner 10

1.3.3 Motion Planner 10

1.4 Computing Architectures to Support Inspection Planning Processes 11

1.4.1 Offline Planning Architecture 11

1.4.2 Online Planning Architecture 13

1.4.3 Combined Planning Architecture 14

1.4.4 The Inspection Planning Framework 14

1.4.5 The Primary Role of Each Process in the IPF 15

1.5 Thesis Problem, Scope, Outline and Contributions 17

1.5.1 Submarine Tank Inspection Planning Problem 17
1.5.2 Thesis Scope 20
1.5.3 Thesis Outline 21
1.5.4 Original Contributions 23
1.6 Chapter Summary 25

Chapter 2 Coverage Path Planning for Autonomous Tank Inspection Robots: A Literature Review 26

2.1 Introduction 26
2.2 Review of Path Planning Algorithms 27
2.2.1 Graph-based Search over Decomposed Environments 28
2.2.2 Sampling-based Path Planning 34
2.2.3 Path Replanning Strategies 38
2.2.4 Path Planning Summary 44
2.3 Review of Coverage Path Planning Algorithms 45
2.3.1 Coverage Planning using Environmental Decompositions 47
2.3.2 Sampling-based Coverage Path Planning 50
2.3.3 Online Coverage Path Planning Approaches 55
2.3.4 Coverage Path Planning Summary 60
2.4 Consolidation of the Literature to Resolve the STIPP 60
2.5 Gap Statement 64
2.6 Chapter Summary 65

Chapter 3 Formulation of a Solution for the STIPP 66

3.1 Introduction 66
3.2 Selection of an Appropriate Coverage Planner to Solve the STIPP 67
3.3 Offline Sampling-Based Coverage Path Planning Using Redundant Roadmaps 69
3.3.1 Solving the Coverage Sampling Problem 70
3.3.2 Solving the Set Cover Problem 74
3.3.3 Solving the Multi-goal Planning Problem 74
3.4 Algorithmic Concerns 75

3.4.1 Uncertainty of the Mesh, Sensing and Motion Models 76
3.4.2 Applicability of a Decoupled Solver for Non-holonomic Robotic Platforms 76
3.4.3 Complex Geometry that Invalidates Complete Coverage 77
3.4.4 Long Computational Time Associated with LPP 77
3.5 Online Adaptation of the Offline Sampling-based Coverage Planner to Resolve the Online Requirements of the STIPP Criteria 78
3.5.1 Strategy 1: Full Replan 80
3.5.2 Strategy 2: Plan Repair 81
3.5.3 Considerations of Replanning Strategies 84
3.6 Limiting Computational Effort by Reducing Extrinsic Planning Constraints 85
3.6.1 Assumptions Placed on the Relationships between the Environment, Mapping System and Adaptive Coverage Planner 86
3.6.2 The Separation of High-fidelity Motion Planning from the Coverage Planner 89
3.6.3 Simplifying Visibility Constraints to Enable High-Fidelity Motion Plans to Achieve Simplified Viewing Locations 92
3.6.4 Summary of Assumptions between the Coverage Planner and the Motion and Visibility Constraints 94
3.7 Chapter Summary 94

Chapter 4 Evaluating the Functionality of the Offline Sampling-Based Coverage Planner in Preparation for Online Planning 97

4.1 Introduction 97
4.2 Implementation of the Existing Offline Sampling-based Coverage Planner 98
4.2.1 Constructing a Redundant Roadmap 99
4.2.2 Handling Trapped Configurations due to *Prison Cells* 102
4.2.3 Introducing a Primitive Rejection Limit to Avoid Over-Sampling Unobservable Primitives 103
4.2.4 Substitution of the TSP Solver for Large Covering Sets 104
4.2.5 Motion Planning and Handling Unachievable Paths 104
4.3 Planning Scenarios to Evaluate Algorithmic Properties 105

4.3.1 Empty Box Environments 106
4.3.2 House Plans Containing a Single Geometrical Change 106
4.3.3 Representative Submarine Tank Environments 107
4.4 Benchmarking the Offline Sampling-based Coverage Planner 109
4.4.1 Computational Observations and Results 109
4.4.2 Discussion 122
4.5 Chapter Summary 124

Chapter 5 Minimising the Variability of the LPP for Stable Online Solutions 126

5.1 Introduction 126
5.2 Removing the $O(n^2)$ Complexity as a Factor of Computational Variability 126
5.3 Variable Planning Iterations 128
5.4 Reasons Why the TSP Solver Produced Variable Solutions 135
5.5 Relaxing the Equality Termination Condition to Compensate for Near-Optimal TSP Solutions Generated by Approximation TSP Solvers 137
5.6 Coverage Planning with the Additional Termination Conditions 140
5.6.1 Computational Observations and Results 140
5.6.2 Discussion 145
5.7 Analysis of the LPP Planning Data for Online Implementation 147
5.7.1 Path Evaluations and Environmental Changes 147
5.7.2 Impact on Online Replanning Strategies 150
5.8 Chapter Summary 150

Chapter 6 Factoring Environmental Influences for Informed Path Planning 152

6.1 Introduction 152
6.2 Underestimating the Connectivity between Configurations 153
6.3 Constructing a Suitable Heuristic 157
6.4 Extracting an Informed Graph through Environmental Properties 158
6.4.1 Configuration-based Graphs 158
6.4.2 Geometric and Topological Graphs that Represent Free Space 159
6.4.3 Skeleton-heuristic Proposal 161

6.5 Constructing an Adjacency Matrix Using the Skeleton-heuristic 162
6.5.1 Connecting Configurations to the Skeleton during the Thinning Process 163
6.5.2 Computation of the Skeleton-heuristic 164
6.5.3 Limitations to the Proposed Skeleton-heuristic Implementation 165
6.6 Coverage Planning using the Skeleton-heuristic 166
6.6.1 Computational Observations and Results 166
6.6.2 Discussion 170
6.7 The Hybrid-heuristic 176
6.8 Coverage Planning with the Hybrid-heuristic 177
6.8.1 Computational Observations and Results 177
6.8.2 Discussion 182
6.9 Application of the Hybrid-heuristic over a Life-like Office Space 183
6.10 Applicability of the Offline Planning Results to Online Planning Proposals to Solve the STIPP 186
6.11 Chapter Summary 188

Chapter 7 Adaptive Sampling-Based Coverage Planning using a Plan Repair Strategy 190

7.1 Introduction 190
7.2 Transitioning from an Offline to Online Planning Problem 191
7.2.1 Defining a New Set System Relationship 193
7.2.2 Defining a ROI within the Configuration and Primitive Spaces 194
7.2.3 Compromised Robot Configuration and Occluded Primitives 197
7.2.4 Environmental and Mapping Influenced Prison Cells 197
7.3 The Adaptive Sampling-based Coverage Planner 200
7.4 ROI Validation 201
7.4.1 Computation of the ROI 203
7.4.2 Revaluating the Trapped Configuration Heuristic due to Partial Map Updates 203
7.5 Online CSP 205
7.5.1 Optimal Sampling Procedure 205

7.5.2 Tracking Occluded Primitives over the Lifetime of the Inspection Task 208
7.6 Online MPP 208
7.6.1 Solving Each ROI Sub-plan 210
7.6.2 Enabling the LPP to Remove Trapped Configurations due to Prison Cells 212
7.7 Representative Concept Demonstration 213
7.8 Probabilistic Completeness of the Adaptive Sampling-based Coverage Planner 215
7.9 Chapter Summary 217

Chapter 8 Computational Analysis of the Adaptive Sampling-Based Coverage Planners 219

8.1 Introduction 219
8.2 Experiment Methodology 220
8.2.1 Adaptive Coverage Planning Using a Full Replan Strategy 221
8.2.2 Experiment Design 222
8.2.3 Planning Environments 223
8.2.4 Simulated Map Updates and Robot Motion 225
8.2.5 Initial Offline Tours 227
8.2.6 Contextualised Planning Outcome: Tour Time and Relative Computational Effort 228
8.2.7 The Selection of a Suitable Redundancy for Online Planning Using the Optimal Sampling Procedure to Create the Redundant Roadmaps 229
8.2.8 Inclusion of the Additional Termination Conditions and Hybrid-heuristic into the Adaptive Coverage Planners 234
8.3 Experiment 1: Size and Scalability of AD-P within a Controlled Environment with Full Map Updates 235
8.3.1 Computational Observations and Results 236
8.3.2 Discussion 253
8.3.3 Summary 260
8.4 Experiment 2: Adaptive Coverage Planning with Full Map Updates in the Representative Tank Environments 261

8.4.1 Computational Observations and Results 262
8.4.2 Discussion 269
8.4.3 Summary 271
8.5 Experiment 3: Simulated Coverage Planning with Partial Map Updates 272
8.5.1 Computational Observations and Results 273
8.5.2 Discussion 282
8.6 Chapter Conclusions and Summary 286
Chapter 9 Summary, Original Contributions, Conclusions, and Future Work 291
9.1 Thesis Summary 291
9.2 Original Contributions 298
9.3 Conclusions 300
9.4 Future Work 306
9.4.1 Additional Termination Conditions: Terminating upon the First Sign of No Improvement 307
9.4.2 Improving the ability of the Hybrid-heuristic to Solve the Multi-Goal Planning Problem Faster 308
9.4.3 Expanding the Capability of the Plan Replan Strategy 311
9.5 Concluding Statement 318
Appendix A Software Packages used to Create Coverage Planners and Heuristics 320
Bibliography 322

List of Figures

Figure 1-1: Collins Class Submarine (ASC Pty Ltd, n.d; reproduced with permission). 2

Figure 1-2: Submarine ballast tanks are empty when the vessel is surfaced (a) and take in sea water to submerge (b) (ASC Pty Ltd, n.d; reproduced with permission). 2

Figure 1-3: Examples of rail-guided inspection platforms for tank inspection. (a) Autonomous Rail-guided Tank Inspection System (ARTIS) (University of Bremen, n.d; reproduced with permission). (b) DORIS (Republished with permission of Offshore Technology Conference, from Nunes et al., 2013; permission conveyed through Copyright Clearance Center, Inc.). 6

Figure 1-4: Examples of legged platforms. (a) CSIRO Magnapod (CSRIO, 2016; © Copyright CSIRO Australia, reproduced with permission). (b) Climbing RObot Caterpillar (CROC; Ward et al., 2014; © UTS Centre for Autonomous Systems, reproduced with permission). 7

Figure 1-5: Concept demonstrator. (a) The platform is equipped with a Velodyne VLP-16 Puck-LITE lidar for mapping and does rely on batteries as power is supplied by an electrical tether. (b) Close-up of the underside of the central leg where the Basler Dart cameras are located. 9

Figure 1-6: Differences between *offline* and *online planning architectures*. (a) An *offline planning architecture*. (b) An *online planning architecture*. The combination of both an *offline* and *online planning architecture* creates a *combined planning architecture* that enables an *online planning architecture* to be initialised with an understanding of the planning problem that was calculated by an *offline planning architecture* before execution. The executive is responsible for guiding the robot successfully to achieve its mission. 12

Figure 2-1: Various paths exist through a given environment. The choice of path is dependent upon the application. 28

Figure 2-2: Path planning using approximate cellular representations. (a) A regular sized grid. (b) Quadtree. 30

Figure 2-3: Various illustrative examples of different forms of environmental decomposition algorithms. (a) A *visibility graph* uses the objects line-of-sight to form a geometric-based graph. (b) A *Voronoi graph* is generated equidistant between objects in the environment. (c) A *navigation mesh* made from different sized polygons. The graph of the mesh is highlighted in blue. 32

Figure 2-4: *Dubins curves* applied over a channel found over the *navigation mesh* created by the *Delaunay Triangulation* to find a path that directs the autonomous vessel through the navigational buoys (green and red) (Wheare et al., 2019; © 2019 IEEE, reproduced with permission). 34

Figure 2-5: Examples of sampling-based path planning algorithms, (a) a *probabilistic roadmap* (PRM) and (b) a *rapid-exploring random tree* (RRT). The shaded red areas highlight narrow regions that samples were not generated within. Sampling within these regions may have resulted in shorter paths to the goal. 37

Figure 2-6: Examples of the three replanning strategies. (a) The new changes obstruct the existing path. (b) A *full replan strategy* replans from the point of detecting an obstacle to the goal position. (c) A *partial replan strategy* creates a new plan from the obstruction to the goal. (d) A *plan repair strategy* only resolves segments of the tour that have been impacted by change. 40

Figure 2-7: An illustration of an abstracted graph that HPA* would produce over the example planning environment. Solving a path requires first solving the abstracted graph to identify which *clusters* (6x6 cells) to solve using A*. 41

Figure 2-8: Example between coverage acquired over the edges of a plan by sampling vs coverage acquired over the edges of a plan with continuous sweeping trajectories. The red fans indicate the coverage of the surface by the robot (blue). 47

Figure 3-1: Different approaches to replan a piecewise inspection plan to changing conditions. (a) The current plan is compromised due to the detection of a new obstacle. (b) *Full replan strategy*. (c) *Partial replan strategy*. (d) *Plan repair strategy*. 69

Figure 3-2: The state-flow diagram demonstrating the offline sampling-based coverage planning algorithm presented (Englot, 2012; reproduced with permission). 70

Figure 3-3: An offline coverage plan solved over a representative 1-metre I-beam structure using the offline sampling-based coverage planner. 71

Figure 3-4: The set system P, Q and the relationship between the continuous configuration space Q and discrete primitive space P. The set system highlights the mapping of a configuration qj to a subset of primitives $pi \subset P$. 72

Figure 3-5: Generating configurations. (a) Configurations are drawn within the local neighbourhood that are defined by the envelope of the sensor parameters. (b) The coverage is calculated, when the sensor parameters are projected onto the surface. (c) The primitive along with neighbouring primitives within the field of view of the sensor become observed. (d) Within this envelope the redundancy can be met by other samples. 73

Figure 3-6: Identifying new features within a partially known environment. (a) A robot (black) has three unknown features within the *a priori* environment. (b) After a full LIDAR scan (red), the mapping system detects the partial outline of the three features that do not correlate to the expected surfaces (white). (c) The *mapping system* registers the changes and appends the partial information known about these features into the map (green). (d) When the robot moves forward new information is gathered about these features. The partial features will become whole once the robot explores more of the environment. 79

Figure 3-7: The proposed *adaptive sampling-based coverage planner* using a *path repair strategy* inside the *Inspection Planning Framework* (IPF). The *adaptive coverage planne*r generates a *region of interest* (ROI) to segment the current plan into *preserved* and *unpreserved segments*. *Unpreserved segments* are replanned to cover the new features and are merged back into the current plan. The updated plan is handed to the motion planner to update the respective motion plans for the segments of the tour that were replanned. This process continues for each map update until the inspection plan is complete. 82

Figure 3-8: Bounding new features in a *region of interest* (ROI). (a) When the mapping system detects change, a ROI can be produced around each detected feature. (b) When the robot moves through the environment and gains more information about the

environment, new ROIs encompass the new changes. 83

Figure 3-9: A spherical collision model is used to represent the ankle joint of a 3-DOF leg on a 18-DOF PhantomX robot containing the camera for inspection. The spherical model is given 6-DOF to find appropriate viewing locations. PhantomX MKIII model adapted from Trossen Robotics (2019). Reproduced with permission. 92

Figure 3-10: Representation of the actual and relaxed visibility constraints. (a) Coverage under rectangular projection constraints restrict the roll of the camera. (b) Assuming a circular projection that underestimates the rectangular projection allows for the calculated coverage to be viewed by any roll angle. 93

Figure 4-1: Transforming a configuration sampled in the spherical frame to the world frame. (a) A configuration is sampled in the spherical frame based on the constraints of the sensor. (b) Transformation of the configuration to the world frame can be achieved by rotating and offsetting the configuration around a primitive normal in the world frame. (c) The result of the transformation angles the configuration to the centre of the selected primitive. 100

Figure 4-2: Visualisation of how the visibility checks were undertaken. (a) A radial nearest neighbour search finds the primitives within the neighbourhood of the configuration based on the maximum Field of Depth (FOD) specified. (b) Primitives within the local neighbourhood are pruned to the Field of View (FOV) of the sensor. 101

Figure 4-3: Calculation of a configuration's coverage. 101

Figure 4-4: Identifying trapped configurations within *prison cells*. (a) A basic example of a prison cell geometry trapping a configuration inside an object prohibiting a feasible path to other configurations. (b) The number of intersections each ray trace makes with interior objects and the mesh boundary determines if a configuration is within free space. 103

Figure 4-5: Planning environments *2x2m* with *6x6m*. 106

Figure 4-6: Planning environments *House* and *House-W*. 107

Figure 4-7: Planning environments *Tank* and *Tank-P4*. 108

Figure 4-8: Sampling configurations around complex geometries such as I-beams and pipe brackets can be challenging in clustered areas. 108

Figure 4-9: Generated coverage plans for *2x2m* and *6x6m.* 112

Figure 4-10: Generated coverage plans for *House* and *House-W*. 113

Figure 4-11: Generated coverage plans for *Tank* and *Tank-P4*. 114

Figure 4-12: Overall planning times for *6x6m*. 116

Figure 4-13: Overall planning times for *Tank*. 116

Figure 4-14: Overall planning times for *Tank-P4*. 116

Figure 4-15: Overall planning times for *House-W*. 117

Figure 5-1: Lazy point-to-point planning data for a trial of 6x6m. 129

Figure 5-2: Lazy point-to-point planning data for a trial of *Tank*. (a) All planning data. (b) A closer examination of the planning data. 132

Figure 5-3: Lazy point-to-point planning data for a trial of *Tank-P4*. (a) All planning data. (b) A closer examination of the planning data. 133

Figure 5-4: Flowchart of how the *additional termination conditions* are integrated into the LPP. 139

Figure 5-5: There are a significant number of RRT paths being evaluated that are not retained in the final solution. 149

Figure 6-1: The evolution of a *lazy point-to-point planner* solution for *House-W*. 155

Figure 6-2: A simplified example demonstrating why the *Euclidean assumption* is an unsuitable metric in *House-W*. The colours in (b) and (c) represent the estimation of the connectivity from the perspective of the starting location. 156

Figure 6-3: Evolution of the Lee et al. (1994) thinning algorithm producing a skeleton within a rectangular space around a central structure. Each image demonstrates how the space is iteratively eroded away from all surfaces (black) until a thin 1D skeleton remains between the sides and the centre object (Wheare, 2018, reproduced with permission). 160

Figure 6-4: Using a skeleton of the environment will allow a graph to be formed that allows the connectivity between configurations to better represent the actual distance, thus aiding the removal of unnecessary path evaluations from occurring. The colours are used to represent the connectivity of the magenta configuration to the other configurations in the planning problem. Green represents a close connection and red represents a distant connection. 161

Figure 6-5: Resultant skeletons (black) of *House-W* using the (a) traditional Lee et al. thinning algorithm without the extra voxel deletion criteria and (b) connecting configurations (yellow) to the main skeleton with the extra voxel deletion criteria. As the walls are hollow, a skeleton is also generated within them. For clarity, these skeletons are removed from the image. 164

Figure 6-6: The final tour generated by the *lazy point-to-point planner* initialised with a *skeleton-heuristic* and *Euclidean assumption* for *2x2m*. The *skeleton-heuristic* increased planning times and tour length for *2x2m* due to the calculation of the heuristic and the overestimation of local connectivity respectively. 171

Figure 6-7: The final tour generated by the *lazy point-to-point planner* initialised with a *skeleton-heuristic* and *Euclidean assumption* for *House-W*. The *skeleton-heuristic* reduced planning times for *House-W* but due to the overestimation of local connectivity tour lengths were longer than tours solved under the *Euclidean assumption*. 171

Figure 6-8: Example scenarios of the overestimation in non-obstacle filled and obstacle filled environments. (a) The representative skeletons of each environment with attached configuration branches. (b) The estimated distances recorded by the skeleton can overestimate local connectivity between configurations. (c) A line-of-sight check between configurations restores the *Euclidean assumption* for configurations not separated by an obstacle. This creates a hybrid distance metric that better approximates the connectivity between the configurations within various types of environments. 173

Figure 6-9: Examples of skeleton branches overestimating connectivity between adjacent configurations (red) for (a) *2x2m* and (b) *House-W*. 174

Figure 6-10: The open space of *Tank* creates overestimation of the skeleton. (a) The resultant skeleton of *Tank*. (b) The centeredness of the skeleton in open space promotes overestimation due to the skeleton branches (highlighted in red). 175

Figure 6-11: The introduction of the *hybrid-heuristic* increases the number of paths being retained in the final solution. 181

Figure 6-12: Planning with the *hybrid-heuristic* over a large complex environment. (a) The resultant skeleton of the office floor space with connected configurations. (b) The final tour. Grey primitives on the floor of the office in (b) are due to anomalies in the mesh

model that made these primitives unobservable. 185

Figure 7-1: Conceptual diagrams of the *adaptive sampling-based coverage planner* implementing a *region of interest* (ROI) to bound the replanning effort to construct the plan repair strategy. (a) The offline plan over a known environment. (b) A ROI bounds the new primitives of the map update. (c) The original positions are validated against the ROI. (d) The *covering sampling problem* (CSP) and *set cover problem* (SCP) produce new configurations to cover all primitives in the ROI. (e) The MPP produces a new sub-plan that is connected back into the existing plan. (f) The plan is passed to the robot and will remain as the only plan until a new map update is provided. 192

Figure 7-2: An updated set system relationship between the configuration space Q and the primitive space M for the online system. Each set is represented by a colour. The colours used for each set will be used to represent the same set in subsequent figures throughout this chapter. 194

Figure 7-3: Generating the *region of interest* (ROI). (a) A single primitive produces its own ROI (ROI_k) based on the maximum field of depth (FOD_{max}). b) The collective of multiple primitives creates the conceptual ROI that is used limit the replanning effort over the environment. 195

Figure 7-4: A conceptual representation of a region of interest (ROI) used to create Q^{ROI} and P^{ROI}. (b) The introduction of P^* creates Q^{ROI} within Q to segment T. (c) After segmenting T using Q^{ROI}, P^{ROI} can be found through including all of Q^{ROI} coverage. The example highlights all $P^{\#}$ within P^{ROI} to ensure all primitives of P^{ROI} remain covered**.** 197

Figure 7-5: Evolving structures in the environment create either an *environmental* or *mapping prison cell* that will invalidate the existence of a path to surrounding configurations outside the cell. As these prison cells are difficult to identify during execution, the unobservable primitives trapped by these either of these prison cell types will be treated equally by the *adaptive coverage planner*. 198

Figure 7-6: Flowchart representation of the *adaptive sampling-based coverage algorithm* using a *plan repair strategy*. The *adaptive sampling-based coverage planner* includes an initial validation step that determines if the existing configurations reside inside or outside the *region of interest* (ROI). The separation allows the *adaptive coverage planner* to only replan regions within ROIs. 201

Figure 7-7: Passing the existing tour through the *region of interest* (ROI) Validation phase. (a) A segment of the current tour. (b) The new primitives (light blue) bounded by the ROI (red). (c) The current tour is evaluated to determine the current status in and around the ROI. 202

Figure 7-8: Revisiting the trapped configuration problem. (a) A complete environment model can handle the strict ray tracing constraint. (b) Relaxing the intersection count constraint enables configurations to exist within partially constructed environments. (c) Clustering around the entrances of an object can cause the Nearest Neighbours Heuristic to pass despite the robot being physically unable to pass through the gap of the object. 204

Figure 7-9: Sampling new configurations across the unobserved primitive set C. A new *redundant roadmap* is generated over the unobserved primitives (blue) within the *region of interest* (ROI). The two existing configurations (yellow) are retained for their unique coverage. A single primitive is partially occluded by the new structure (red) and is excluded from the coverage. 206

Figure 7-10: Optimal Sampling Procedure. (a) Sampling above the surface at the *optimal position* allows for (b) a better observation of the surface than a large percentage of random samples, especially below the *optimal position*. 207

Figure 7-11: Replanning path segments back into existing plan. (a) The path segment is replanned between the entry and exit gates into around the region of interest. (b) The tour and map are updated to reflect the new paths and coverage determined in the current iteration. The next iteration will exclude the unobserved primitives completely (red) if they are not included within the *region of interest*. 209

Figure 7-12: The Travelling Salesman Sub-Problem. (a) A standard *travelling salesman problem* (TSP) solution starting and finishing at the entry gate. (b) A solution using a dummy configuration to solve the TSP sub problem. (b-i) Assigning a zero-edge cost from the start and finish to the dummy configuration (black) with a large edge cost from the dummy configuration to all other configurations (blue). This value needs to be greater than the largest edge cost in the problem (red) to avoid connecting any other configurations to the dummy configuration (b-ii) The final solution forces the dummy configuration to be adjacent to the gate pair (orange). (b-iii) Removing the dummy configuration orders a TSP Sub-Problem starting and finishing at the gate pair. 211

Figure 7-13: Example of a trapped configuration within a partially constructed object. RRTs attempt to connect to configurations 1-2 and 2-3 however collision checks on RRT paths on the entry of the object (red) impede these paths from connecting to the trapped configuration (2). Configuration 2 is deemed to be trapped and is removed from the tour. 212

Figure 7-14: The adaptive sampling-based coverage planner replanning over a representative I-Beam structure. (a) Offline plan. (b) Introduction of a new structure bounded within an approximated *region of interest* (ROI). (c) ROI Validation. (d) Online covering sampling problem (Online CSP). (e) Online multi-goal planning problem (Online MPP). (f) Updated inspection plan. 214

Figure 8-1: Flowchart representation of *adaptive sampling-based coverage planner* using the *full replan strategy*. 222

Figure 8-2: Controlled simulation environments and update order. (a) The smallest *2x2m* environment containing five changes (green). (b) The largest planning environment *8x8m* with the maximum nine changes. (c) The order in which each update will appear in the experiments. The green and red dots indicate the designated start and end positions for every plan respectively. 226

Figure 8-3: *Tank* and the additional pipe network to construct *Tank-P4.* 226

Figure 8-4: *Optimal Sampling Procedure* (OSP) over *2x2m*. (a) Covering set sizes of varying redundancies using the OSP against a fully random roadmap construction. (b) Number of *optimal* and random samples taken in the final tour for a *redundant roadmap* constructed by the OSP. 231

Figure 8-5: *Optimal Sampling Procedure* (OSP) over *8x8m_9.* (a) Covering set sizes of varying redundancies using the OSP against a fully random roadmap construction. (b) Number of *optimal* and random samples taken in the final tour for a *redundant roadmap* constructed by the OSP. 231

Figure 8-6: *Optimal Sampling Procedure* (OSP) over *Tank*. (a) Covering set sizes of varying redundancies using the OSP against a fully random roadmap construction. (b) Number of *optimal* and random samples taken in the final tour for a *redundant roadmap* constructed by the OSP. 232

Figure 8-7: *Optimal Sampling Procedure* (OSP) over *Tank-P4*. (a) Covering set sizes of varying redundancies using the OSP against a fully random roadmap construction. (b)

Number of *optimal* and random samples taken in the final tour for a *redundant roadmap* constructed by the OSP. 232

Figure 8-8: Configurations sampled above the *optimal* viewing location viewing the same primitive are likely to be taken in the set cover due to the extra primitives they might observe. 233

Figure 8-9: Implementation of the *hybrid-heuristic* for each *adaptive coverage planner*. 235

Figure 8-10: Overall planning times for AD-P(R), AD-P and AD-R. 239

Figure 8-11: Relative comparison between overall planning times of AD-P and AD-R. 241

Figure 8-12: Relative comparison between overall planning times of AD-P(R) and AD-R. 241

Figure 8-13: Ratio of each planning process for AD-P(R) as a percentage of the overall planning time. 242

Figure 8-14: Ratio of each planning process for AD-P as a percentage of the overall planning time. 242

Figure 8-15: Ratio of each planning process for AD-R as a percentage of the overall planning time. 242

Figure 8-16: A visual comparison between AD-P, AD-P(R) and AD-R. (a) Initial offline tour. (b) ROI Validation phase. (c) Online CSP phase. (d) Online MPP phase and resultant final tour. 244

Figure 8-17: New configurations introduced into each planning problem for AD-P(R), AD-P and AD-R. 246

Figure 8-18: Number of path evaluations required to solve the multi-goal planning problem for AD-P(R), AD-P and AD-R. 246

Figure 8-19: Difference between the number of configurations removed and introduced to solve a planning problem for AD-P and AD-P(R) for (a) one feature, (b) five features and (c) nine features. 247

Figure 8-20: Despite the influence of the features on the initial tour, a consistent number of configurations were being generated for each feature over each planning environment for all features. (a) Results for AD-P(R). (b) Results for AD-P. 248

Figure 8-21: Number of planning iterations for AD-P(R), AD-P and AD-R. 248

Figure 8-22: The tour degradation of AD-P for the amount of replanning required to perform a tour update. 251

Figure 8-23: The tour degradation of AD-P(R) for the amount of replanning required to perform a tour update. 252

Figure 8-24: Relative *tour time* between AD-P and AD-R. 254

Figure 8-25: *Relative computational effort* between AD-P and AD-R. 254

Figure 8-26: Routing through pre-existing tour locations can degrade tour quality. The dashed lines are the point-to-line comparison used between each paired gates. 259

Figure 8-27: Routing through pre-existing tour locations can degrade tour quality. The dashed lines are the point-to-line comparison used between each pair of gates. 259

Figure 8-28: Overall planning times of each *adaptive coverage planner* over each representative tank environment. 265

Figure 8-29: The time each Online MPP for AD-P(HY) and AD-R(HY) spent creating the *hybrid-heuristic* and solving all *multi-goal planning problem*s. 265

Figure 8-30: Ratio of each planning process for each *adaptive coverage planner* in the representative tank environments as a percentage of the overall planning time. 267

Figure 8-31: Tour length for each *adaptive coverage planner* for the representative tank environments. 268

Figure 8-32: *Relative computational effort* of AD-P against AD-P(HY) AD-R and AD-R(HY). 269

Figure 8-33: Overall planning times and the *average replan time per iteration* (ARTPI) for AD-P and AD-R over all tested environments. 275

Figure 8-34: Speed-up of AD-P over AD-R for the controlled environments. 275

Figure 8-35: Distribution of each planning process as a ratio of overall planning time for AD-P. 277

Figure 8-36: Distribution of each planning process as a ratio of overall planning time for AD-R. 277

Figure 8-37: The additional increase in replanning led to AD-R producing covering set sizes, planning iterations and tour lengths that were larger than AD-P over all planning problems. 278

Figure 8-38: Evolution of inspection plans for AD-P and AD-R for *4x4m_5*. 280

Figure 8-39: *Tour times* and *relative computation effort* (RCE) for *Experiment 3*. 282

Figure 8-40: The average percentage difference between tour lengths produced in *Experiment 1* and *2* against the tour lengths produced in *Experiment 3*. 284

Figure 8-41: The average percentage difference between covering set sizes produced in *Experiment 1* and *2* against the covering set sizes produced in *Experiment 3*. 284

Figure 9-1: Using watersheds to connect configurations to a skeleton. Each shaded region would cluster together groups of configurations to form *a Covering Travelling Salesman Problem*. Watershed over office floor space courtesy of Jonathan Wheare. 310

Figure 9-2: Proposed solution to use topological skeletons and watersheds to create a *Covering Travelling Salesman Problem* (TSP) that can be used to solve smaller *multi-goal planning problems* (MPPs) in large complex planning environments. 310

Figure 9-3: The difference in skeletons produced in free space (a) and a skeleton bound to be constructed within an inflated surface boundary (b). 311

Figure 9-4: Solving the *hybrid-heuristic* first will inform the Online MPP of the adjacency between configurations before sorting the covering set. 314

List of Tables

Table 3-1: Visibility and collision model constraints. 95

Table 4-1: Overall planning times, configuration set sizes and overall tour lengths for each of the planning scenarios. 110

Table 4-2: *Coverage sampling problem* (CSP) and *multi-goal planning problem* (MPP) planning times for each of the planning scenarios. 110

Table 4-3: Coefficient of variation across the planning data for all planning scenarios. 117

Table 4-4: Ratio of Direct and RRT paths used in the final tour. 119

Table 4-5: The difference between minimum and maximum within each set shows no correlation between longer planning times and shorter tour lengths. 120

Table 4-6: List of expected outcomes for each model set and the respective result from the experiment. 121

Table 4-7: Number of planning iterations and paths evaluated to solve each planning problem. 121

Table 5-1: *Lazy point-to-point planner* planning data for all planning problems. 127

Table 5-2: A comparison between best tour length and final tour length highlight there is no significant difference between the two solutions. 131

Table 5-3: The presence of iterations that evaluated no new paths or repeat previous solutions is prevalent throughout every planning problem. 131

Table 5-4: The average number of repeated and unique solutions for each planning problem. 134

Table 5-5: Planning times for all planning scenarios using the *additional termination conditions.* 141

Table 5-6: *Lazy point-to-point planner* attributes for all planning scenarios using the *additional termination conditions.* 141

Table 5-7: A comparison between the termination conditions between trials of both experiments. 142

Table 5-8: Relative performance and percentage difference between trials using the original and *additional termination conditions.* 143

Table 5-9: Statistical and practical significance between trials using the original and *additional termination conditions.* 143

Table 5-10: Lazy point-to-point planning data for all planning problems using the *additional termination conditions*. 144

Table 5-11: Analysis of the number of path types evaluated, taken and retained in the final solution. 149

Table 6-1: Planning times, tour lengths and *lazy point-to-point planner* attributes for all planning scenarios using the *skeleton-heuristic*. 168

Table 6-2: Analysis of the time taken to create the *skeleton-heuristic* and the resultant time to solve the *multi-goal planning problem* (MPP). 168

Table 6-3: Relative performance and percentage difference between trials using the *Euclidean assumption* and the *skeleton-heuristic*. 169

Table 6-4: Statistical and practical significance between trials using the *Euclidean assumption* and the *skeleton-heuristic.* 169

Table 6-5: Planning times, tour lengths and *lazy point-to-point planner* attributes for all planning scenarios using the *hybrid-heuristic.* 178

Table 6-6: Analysis of the time taken to create the *hybrid-heuristic* and the resultant time to solve the *multi-goal planning problem* (MPP). 178

Table 6-7: Relative performance and percentage difference between trials using the *Euclidean assumption* and the *hybrid-heuristic.* 179

Table 6-8: Statistical and practical significance between trials using the *Euclidean assumption* and the *hybrid-heuristic.* 179

Table 6-9: Analysis of the number of path types evaluated, taken and retained in the final solution for trials solved under the *hybrid-heuristic*. 181

Table 6-10: Statistical analysis of the planning times between Euclidean and *hybrid-heuristic* trials. 184

Table 6-11: Time taken to create the *hybrid-heuristic* and the resultant time to solve the *multi-goal planning problem* (MPP). 184

Table 6-12: The resultant planning attributes of the *lazy point-to-point planner*. 184

Table 8-1: Resolution and primitive counts for controlled and representative environments. 226

Table 8-2: Offline random and raster coverage plans to initialise the *adaptive coverage planners.* 228

Table 8-3: Statistical analysis of AD-P(R), AD-P and AD-R. 237

Table 8-4: Statistical analysis between planning attributes of the AD-P and AD-R initialised with the *Euclidean assumption* and *hybrid-heuristic.* 263

Table 8-5: Relative performance and statistical significance between planning attributes between all *adaptive coverage planners*. 264

Table 8-6: The number of trials for each planning problem conducted for *Experiment 3*.273

Table 8-7: Statistical analysis for *Experiment 3*. Shaded *tour times* indicate the robot will wait for tour updates. 274

Glossary

2.5D	Two-and-a-half dimensional
2D	Two-dimensional
3D	Three-dimensional
AABB	Axis-Aligned Bounding Box
AD-P	Adaptive Sampling-based Coverage Planner using Plan Repair Strategy
AD-P(HY)	AD-P initialised with the Hybrid-heuristic
AD-P(R)	AD-P initialised with the Raster Tour
AD-R	Adaptive Sampling-based Coverage Planner using Full Replan Strategy
AD-R(HY)	AD-R initialised with the Hybrid-heuristic
ARC	Australian Research Council
ARTPI	Average replan time per iteration
AUV	Autonomous Underwater Vehicle
BTL	Best Tour Length
CLK	Chained Lin-Kernighan
CPP	Coverage Path Planning
CSIRO	Commonwealth Scientific and Industrial Research Organisation
CSP	Coverage Sampling Problem
CTL	Current Tour Length
CV	Coefficient of Variation
DOF	Degrees of Freedom
FCD	Full-Cycle Docking
FLANN	Fast Library for Approximate Nearest Neighbours

FOD	Field of Depth
FOV	Field of View
FTL	Final Tour Length
FZEI	First Zero Evaluation Iteration
FZEL	First Zero Path Evaluation Tour Length
GPU	Graphical Processing Unit
IMU	Inertial Measurement Unit
IPF	Inspection Planning Framework
IQR	Interquartile Range
IT	Iteration
LPP	Lazy Point-to-point Planner
MPP	Multi-goal Planning Problem
NBV	Next-Best View
NNH	Nearest Neighbours Heuristic
NP	Non-deterministic Polynomial-time
OMPL	Open Motion Planning Library
Online CSP	Online Coverage Sampling Problem
Online MPP	Online Multi-goal Planning Problem
OPCODE	Optimal Collision Detection
OSP	Optimal Sampling Procedure
PCL	Point Cloud Library
PE	Path Evaluation
PQP	Proximity Query Package
PRM	Probabilistic Roadmap
PTL	Previous Tour Length
QB	Quick-Borůvka
QB-CLK	Quick-Borůvka-Chained Lin-Kernighan
RCE	Relative Computational Effort
ROI	Region of Interest
RRT	Rapidly Exploring Random Tree
R-TCH	Relaxed-Trapped Configuration Heuristic

RTCNDM	Research Training Centre for Naval Design and Manufacturing
SCP	Set Cover Problem
SLAM	Simultaneous Localisation And Mapping
STIPP	Submarine Tank Inspection Planning Problem
TCH	Trapped Configuration Heuristic
TCH-PP	Trapped Configuration Heuristic for Path Planning
TL	Tour Length
TSP	Travelling Salesman Problem
UAV	Unmanned Aerial Vehicle
VPP	View Planning Problem
ZEI	Zero Path Evaluation Iteration
ZETL	Zero Path Evaluation Iteration Tour Length

Nomenclature

$\mathbb{R}^2$	Two-dimensional space
$\mathbb{R}^3$	Three-dimensional space
θ	Polar angle in spherical coordinate system
ϕ	Azimuth angle in spherical coordinate system or FOV_{angle}
r	Range distance in spherical coordinate system
γ	Vector of the sensor's projection angle to primitive normal
ρ	Primitive normal vector
ζ	Resulting angle between ρ and γ
C	Residual coverage remaining after the ROI Validation phase
FOD_{min}	Maximum FOD to observe primitives with quality
FOD_{max}	Minimum FOD to observe primitives with quality
FOV_{angle}	Field of View angle in radians
G	Offline tour / Current tour (Replaced by T for online definition)
k	The number of times each p_i must be viewed (redundancy)
l_s^R	A resolved path segment consisting of an ordered set of collision-free paths between configurations of $q_j \subset Q^{VALID}$
L^R	A set of resolved path segments
l_s^U	An unresolved path segment that consists of an unordered set of configurations of S^*within Q^{ROI}
L^U	A set of unresolved path segments
m_{index}	Threshold index supplied by the mapping system to delineate between P and P^*
M	A map update from the mapping system *s.t.* $M = \{ P \cup P^* \}$

N	A set of configurations that cover P (Replaced by S for online definition used in Online CSP)
p_i	A geometric primitive
P	A finite set of discrete geometric primitives p_i comprising of the existing structure to be inspection
p_k^*	A newly detected geometric primitive
P^*	A finite set of newly detected geometric primitives p_k^* comprising a newly detected structure to be inspected
$P^{\wedge}$	Set of $p_i \subset P$ in ROI
$P^{\#}$	Set of actual $p_i \in P$ influenced by P^*
$P^{OCCLUDE}$	Primitives that become unobservable due to P^*
P^{ROI}	A set of all primitives within *ROI s.t.* $P^{ROI} = \{P^{\wedge} \cup P^*\}$
q_j	A robot configuration
q_n^*	New configurations sampled within *ROI*
q_g^{Entry}	An entry gate into Q^{ROI}
q_g^{Exit}	An exit gate out of Q^{ROI}
Q	The robot configuration space
$Q^{COMPROMISED}$	Set of $q_j \in T$ inside Q^{ROI} that will be removed from T
Q^{Entry}	Set of all entry gates into Q^{ROI}/ ROI
Q^{Exit}	Set of all exit gates into Q^{ROI}/ ROI
Q^{NEW}	Set of new configurations generated to cover C
Q^{ROI}	The robot configuration space within ROI
Q^{VALID}	Set of configurations $q_j \in T$ that are not to be removed for replanning
Q^{VIABLE}	Set of existing configurations $q_j \in T$ in Q^{ROI} that are candidates for replanning in Q^{ROI}
ROI	Region of Interest around P^* that creates P^{ROI} and Q^{ROI}
ROI_k	An individual ROI around p_k^*
s_a^*	A configuration attains coverage of P^{ROI} that needs to be sorted into l_s^U
S	A set of configurations that cover P *s.t.* $S \subset Q$ (Replaces N for online definition)
S^*	Result of SCP over Q^{VIABLE} and Q^{NEW}

$T(G)$	A finite set of robot configurations which comprise the set of goals selected for traversal (tour/plan). T is the ordered set of S that compromises a connection of collision-free paths through Q.
T^*	An updated inspection plan
v	Set of primitives $p_i \subset P$ observed by q_j
V	Set of all primitives covered by Q
v^*	Set of primitives covered by $q_j \in Q^{VIABLE}$ (May include $p_k^* \in P$)
V^*	Set of primitives covered by Q^{VIABLE}

Publications

Publications that have been developed from this work:

Pivetta R., Lammas, A., Short, A., Johanasson, M., Wardle, C., Summat, K. and Van Duin, S. (2017). Submarine Tank Inspection Robots, *4th SIA Submarine Science, Technology and Engineering Conference 2017* (SubSTEC4)

Pivetta, R., Lammas, A. and Sammut, K. (2018). 3D Adaptive Coverage Planning for Confined Space Inspection Robots, *Technology and Science for the Ships of the Future: Proceedings of NAV 2018*, pp. 496-503, doi:10.3233/978-1-61499-870-9-49

Chapter 1

Towards Automated Submarine Tank Inspection

1.1 Motivation

The sustainability of the Australian Navy's fleet of Collins Class Submarines is imperative for the national security and protection of Australian maritime borders and approaches (Figure 1-1; Australian Government: Department of Defence, 2016). To ensure operational readiness, each vessel in the fleet is subjected to a rigorous two-year Full-Cycle Docking (FCD) maintenance service after ten years of deployment (Coles and Greenfield, 2016). During the two-year FCD, manual inspection of the submarine tanks is an important component of the maintenance cycle to ensure their structural integrity during the next ten years of service.

The inspection for corrosion inside the submarine tanks is of utmost importance. Corrosion, left untreated, is one of the major causes of marine structural failures and vessel decommissioning (Gu et al., 2009; De Baere et al., 2013; Heyer et al., 2013). Of all the tanks onboard the submarine, the ballast tanks, which take on sea water to allow the vessel to submerge and vent the water to surface (Figure 1-2), are highly susceptible to aqueous and microbiological corrosion due to the build-up of fouling. These foreign materials wear away the protective paint layer used to limit the impact of corrosion during the service life of the submarine.

Considerable time and effort is required to perform the manual inspection of the ballast tanks. Many hours are spent cleaning and fumigating to rid the ballast tanks of the biofouling and toxins that accumulate over ten years of service before it is deemed suitable for human entry. Personnel are required to have certified training and be equipped with the appropriate personal protective equipment before entering these spaces (Safe Work Australia, 2017).

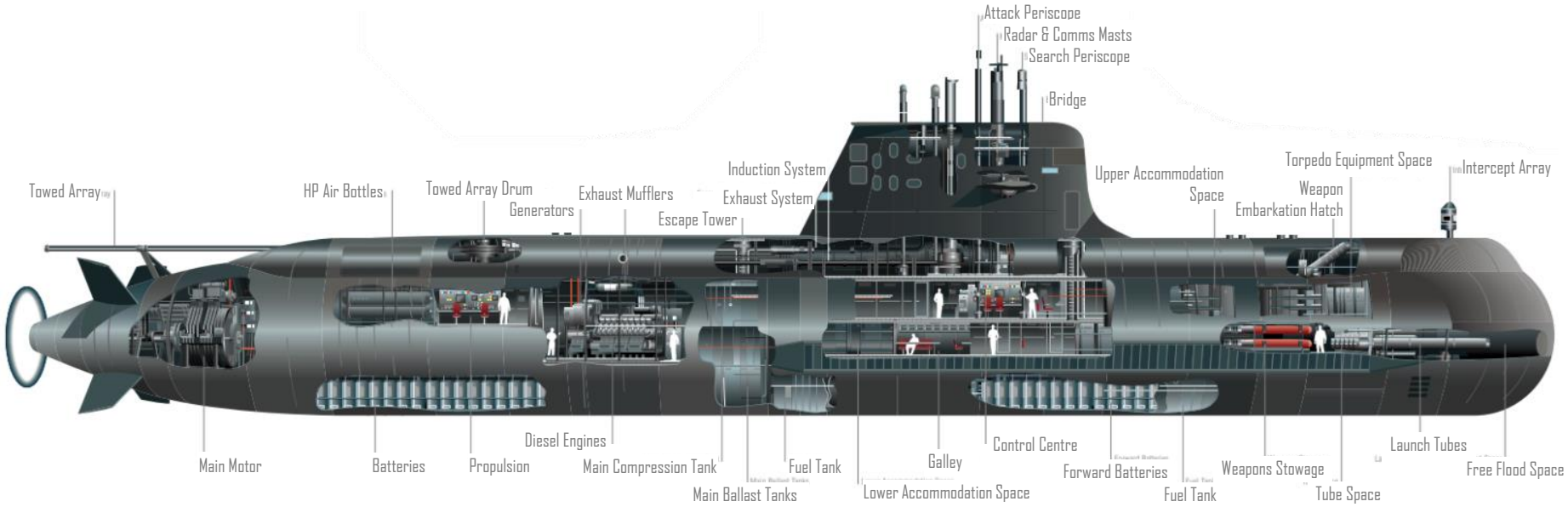
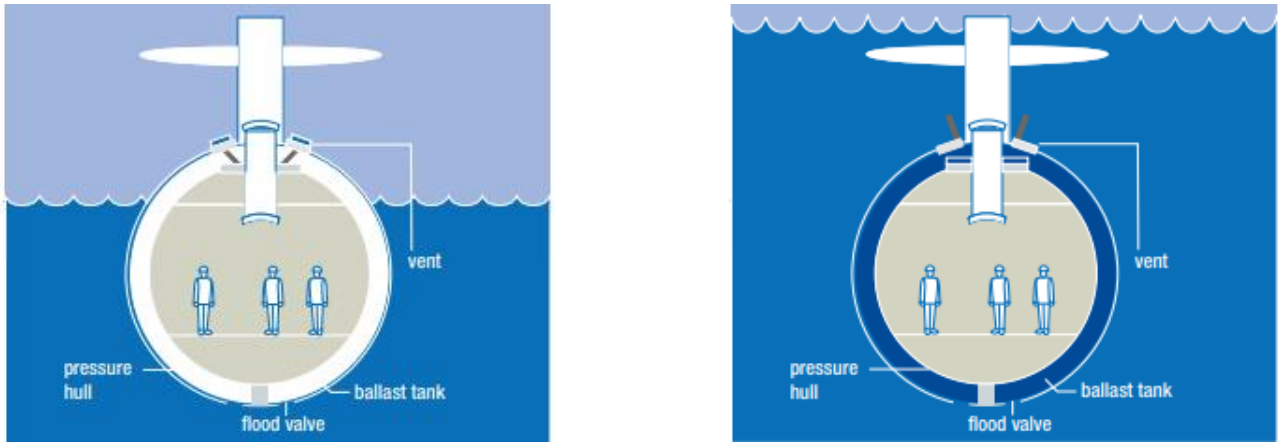


Figure 1-1: Collins Class Submarine (ASC Pty Ltd, n.d; reproduced with permission).



(a) (b)

Figure 1-2: Submarine ballast tanks are empty when the vessel is surfaced (a) and take in sea water to submerge (b) (ASC Pty Ltd, n.d; reproduced with permission).

Tank inspections are carefully scheduled as workers cannot be exposed to the exterior noises generated by perimeter maintenance whilst conducting a tank inspection. Also, with some tanks connected in series, additional spotters are required to ensure the safety of each worker inside each confined space.

The manual inspection of ballast tanks requires significant attention to detail whilst the workers are subject to physically and mentally uncomfortable conditions. Accessible via small manholes, the confined spaces are tight and comprise multiple T and I frame stiffeners and internal pipe networks that make manoeuvring across these steel structures very challenging and demanding for the human body. Inspectors must examine all interior surfaces for evidence of corrosion, paint delamination and potential defects, and in many cases use mirrors to see into areas that their bodies cannot physically fit. Taken together, there is a clear need for an alternative approach which reduces the cost, time, and risk to workers for such a critical component of the FCD.

Autonomous robotic platforms provide the opportunities to aid the inspection of buildings and structures that, given updated work, health and safety regulations, deem confined spaces to be hazardous for humans to inspect (Safe Work Australia, 2017). The advancements in compact sensing capabilities, computational hardware and actuator systems has enabled robotic systems to perform high resolution inspections that;

1) digitally reconstruct buildings and structures (Blaer and Allen, 2009; Klingensmith et al., 2015),
2) survey the quality of road surfaces and tunnels (Montero et al., 2015),
3) inspect defects on power lines (Katrasnik, Pernus and Likar, 2009),
4) survey ship hulls for foreign devices (Hover et al., 2012), and
5) map underwater caves and seabeds at greater depths than divers can safely achieve (Weidner et al., 2017; Carreras et al., 2018).

There are significant benefits to automating the inspection process. First, automating submarine tank inspection, using an autonomous inspection robot platform, would reduce the risks posed to workers by minimising their exposure to the hazardous environment of the ballast tank. Second, by supplying the robot with sufficient power, it can work for longer periods or at time not convenient for human workers. Third, autonomous inspection can proceed irrespective of any perimeter maintenance, enabling the option to reorder the scheduling of ballast tank inspection within the FCD. The fourth point, a robot can digitally

record and update valuable information about the current status of the tanks that can be used as a point of comparison for future FCDs. As the world prepares for the next industrial revolution, Industry 4.0, digital records are going to play an integral part in tracking the lifecycle of a building or structure (Australian Government: Department of Industry, Innovation and Science, 2019).

Equipping an autonomous robot with appropriate sensors, such as cameras and lidar, will enable the robot to objectively record all internal surfaces of the tanks. The information gathered by the robot can be used to generate a digital three-dimensional (3D) representation of the submarine tank environment with a visual photographic overlay. These digital renderings will enable the maintenance engineers to make detailed inspections of the surfaces without entering the tanks.

The tank inspection problem has brought industry and research collaborators together from ASC Pty Ltd, Flinders University and the University of Wollongong in a research project under the auspices of the ARC Research Training Centre for Naval Design and Manufacturing (RTCNDM). The objective of the collaboration was to investigate the possibility of deploying an inspection robot to perform an autonomous inspection over all interior tank surfaces to eliminate the need for human access into such confined spaces. To the research group's knowledge, the automated inspection of submarine tanks had not been achieved with a free-roaming robotic platform, especially not inside the Collins Class submarines. The collaboration group was tasked with the development of an autonomous inspection platform, with the emphasis placed on the algorithms that support the autonomy.

1.2 Robotic Platforms for Confined Space Inspection

Designing a robotic system for performing tasks within a confined space is a challenging problem. Given the complexities of each confined space and the respective application, robotic systems are usually specifically designed for their designated task. Applications of confined space robotics include;

1) visual inspection of marine and ancient structures (Bibuli et al., 2011; Richardson et al., 2013),
2) abrasive blast cleaning (Lee et al., 2010; Sabre Autonomous Solutions, 2019),
3) pipe inspection (Wu et al., 2015; Mills et al., 2018), and
4) maintenance and repair (Dong et al., 2019).

Comprehensive surveys by Shukla and Karki (2013) and Botti, Ferrari, and Mora (2017) provides an insight into a number of platforms used for a range of confined space inspection tasks. This survey confirms the difficulty of designing a platform for the inspection of the Collins Class submarine, as there are strict criteria regarding the types of platforms that may be suitable.

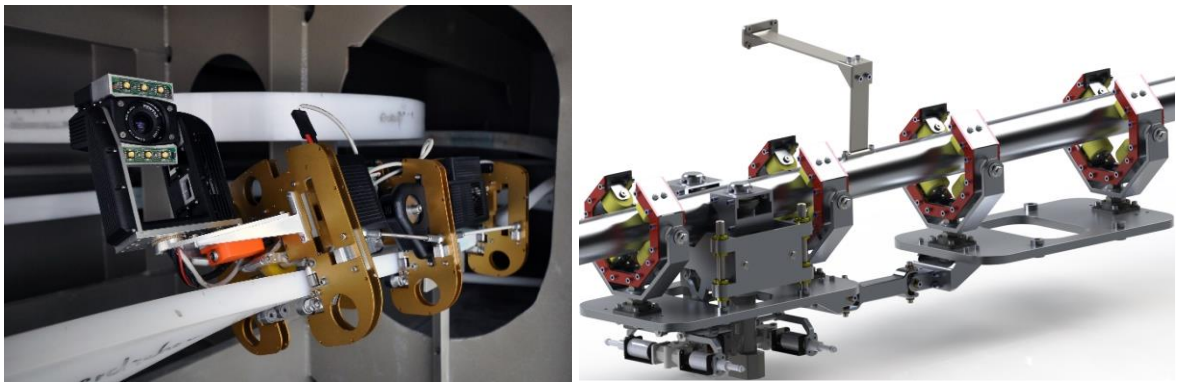
Designing a robotic platform for submarine inspection encompasses the mechanical, electrical and algorithmic complexities required to solve the specific task. For the submarine tank inspection task an appropriate platform is one that;

1) is sufficiently compact and agile to enter and manoeuvre around the confined steel spaces on-board a submarine,
2) has the ability to inspect a variety of different tank layouts and configurations,
3) is able to sufficiently carry all the sensory equipment required to perform a thorough inspection, such as cameras or lidar for visual inspection and object construction,
4) requires no modification or installation of additional structures by humans to support the use of the autonomous platform,
5) has the ability to support a tether to ensure electrical power is not just reliant on batteries for longer inspections, and
6) supports data communication within the tether to ensure the robot is always in constant communication with the host computer and operator.

The determining factor of the effectiveness of a given robotic platform, situated within these tight confined spaces, is defined by the type of locomotion system implemented. A review of suitable locomotion for Collins Class submarines was presented in Pivetta, Lammas and Summat et al. (2017). The strict criteria placed on the types of platforms that are suitable to be placed inside the Collins Class Submarine tanks excludes the following types of robotic platforms from adoption;

1) Rail-guided platforms,
2) Unmanned Aerial Vehicles (UAVs), and
3) wheels or tracked platforms.

Rail-guided platforms (Figure 1-3; Christensen et al., 2011[a,b]; Nunes et al., 2013) require either, temporary or permanent infrastructure, and therefore require human installation. This introduces several design challenges to integrate the associated infrastructure and would require removal before recommissioning. Deploying a small UAV is not practical as it would



(a) (b)

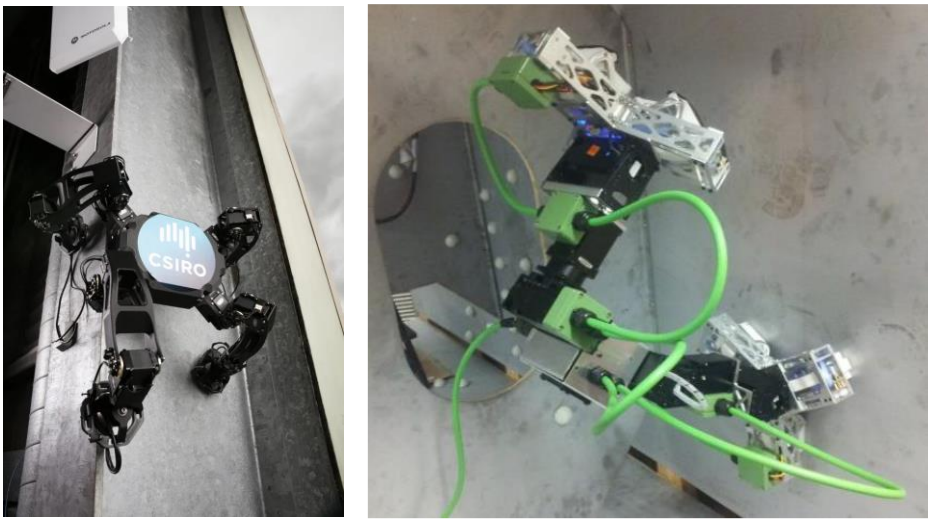
Figure 1-3: Examples of rail-guided inspection platforms for tank inspection. (a) Autonomous Rail-guided Tank Inspection System (ARTIS) (University of Bremen, n.d; reproduced with permission). (b) DORIS (Republished with permission of Offshore Technology Conference, from Nunes et al., 2013; permission conveyed through Copyright Clearance Center, Inc.).

not have the capacity to carry a sufficient payload to accommodate all the necessary sensors required for both inspection and localisation, nor have sufficient power for long inspections.

Additionally, as a Global Positioning System (GPS) signal is not accessible within the tanks, a UAV would have to rely upon external systems, such as motion capture (How et al., 2008), to localise accurately for map registration. Without the capacity to carry a tether for data communication, a UAV would have to rely on WiFi which has the potential to be interrupted or disconnected and has limited data bandwidth. Constant communication to a host computer is essential to track the progress of the robot. Finally, while wheeled or tracked platforms would be suitable to carry the required sensory equipment and be connected via a tether for data communication and electrical power, the structures within each tank variation can be quite different and could pose a challenge for a wheeled platform to manoeuvre around.

1.2.1 Multi-legged Platforms

Pivetta et al. (2017) suggested that a suitable platform for submarine tank inspection is one that is constrained to move along the tank surfaces and possesses the ability to attach or climb over and around the steel frames. The survey indicated that a multi-legged platform with electromagnetic adhesion at the ends of each leg fulfils the criteria for a self-contained platform that possesses the ability to manoeuvre across uneven surfaces. Despite being constrained to move along the surfaces it is inspecting, a multi-legged platform is holonomic, having the ability to move freely in any direction over the surface, and with electromagnetic feet the platform can carry the required sensory payload and tether.



(a) (b)

Figure 1-4: Examples of legged platforms. (a) CSIRO Magnapod (CSRIO, 2016; © Copyright CSIRO Australia, reproduced with permission). (b) Climbing RObot Caterpillar (CROC; Ward et al., 2014; © UTS Centre for Autonomous Systems, reproduced with permission).

Figure 1-4 illustrates two different examples of multi-legged platforms that possess electromagnetic adhesion. Both the Commonwealth Scientific and Industrial Research Organisation (CSIRO) Magnapod (CSRIO, 2016; Figure 1-4a), designed for ballast tank inspection, and the Climbing RObot Caterpillar (CROC; Ward et al., 2014; Figure 1-4b) designed for inspection within the Sydney Harbour Bridge, possessed at least three *degrees of freedom* (DOF) per leg to allow enough dexterity to manoeuvre within the respective environments.

In general, multi-legged platforms with at least 3-DOF per leg allows enough dexterity to manoeuvre in an uneven environment and enough redundancy in the number of legs that are in contact with the surface at any one time. The 3-DOF per leg enables the legs to articulate in a large number of possible poses. Both these requirements are essential for a platform to manoeuvre in an uneven environment and to allow the platform to climb, particularly in the transition from horizontal to vertical surfaces.

While a multi-legged platform may possess the ability to manoeuvre freely in the confined spaces of a ballast tank, it is the electromagnetic adhesion that enables the platform to climb within these steel environments. For this submarine inspection task, electromagnetic adhesion is preferred over pneumatic adhesion. Pneumatic adhesion has the potential to damage the existing surface paint and requires time to create a tight seal to connect to the surface reliably and effectively (Silva, Machado and Tar, 2008; Brusell, Andrikopoulos and Nikolakopoulos, 2016). Surfaces also need to be clean to ensure air-tight contact. As there

is no guarantee all the surfaces can accommodate pneumatic adhesion, it is therefore not suitable for this application. The benefit of electromagnetic adhesion is that it has the ability to be systemically activated and deactivated, whereas permanent magnets require a force to remove them from the metal surface.

Of the designs reviewed, there was no multi-legged platform that could meet all the requirements for immediate deployment inside Collins Class submarine ballast tanks. The most amenable platform was the CSIRO Magnapod. However, with a limited sensory capability, the Magnapod does not currently possess the autonomous capability to perform the required inspection. At the time of writing this thesis, these platforms are still very much in the research and development stage and are not commercially available. As a suitable platform had been identified, a concept demonstrator was developed based off the findings of the review with the sole focus of testing the algorithmic capability required to perform an autonomous inspection.

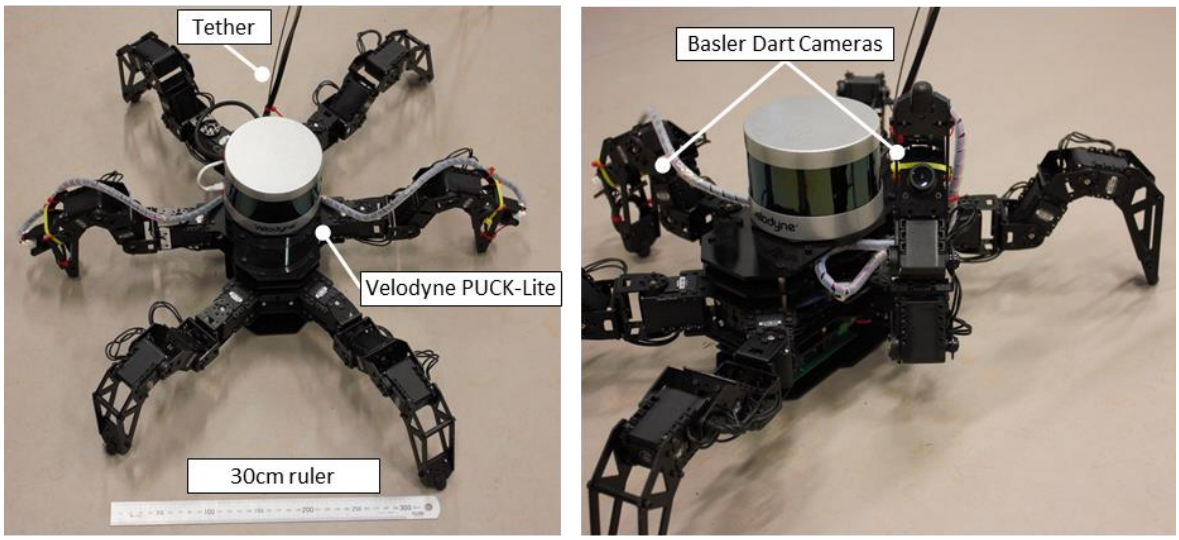
1.2.2 Concept Demonstrator

The concept demonstrator that is the testbed for the developed algorithms is based on the PhantomX Mark III hexapod robot from Trossen Robotics (Figure 1-5; Trossen Robotics, 2019). Equipped with 18 Dynamixel AX-18A servo motors, the PhantomX has 18-DOF allowing the robot to transverse complex terrain with individual leg control.

To enable mapping and localisation, the hexapod was fitted with a Velodyne VLP-16 Puck-LITE lidar. The positioning was additionally aided with an inertial measurement unit (IMU). To perform close-up inspections, two of the robot legs are equipped with Basler Dart cameras. Both cameras are placed on the central legs on the underside of the tibia joint (Figure 1-5b). Placing the cameras on the underside of the tibia joint allows for a greater range compared to placing the cameras on the front of the tibia. Having two cameras enables the robot flexibility to choose which camera is best to acquire the image, given its current position, to avoid having to continually rotate the platform to accommodate just one camera.

The multi-legged platform also contains a tether to continually supply electrical power and provide image data to the host computer during the inspection. The benefit of using a tether allows for emergency recovery, whereby if the robot malfunctions, it can be easily located and recovered.

This initial concept demonstrator does not possess the electromagnetic adhesion upon each



(a) (b)

Figure 1-5: Concept demonstrator. (a) The platform is equipped with a Velodyne VLP-16 Puck-LITE lidar for mapping and does rely on batteries as power is supplied by an electrical tether. (b) Close-up of the underside of the central leg where the Basler Dart cameras are located.

of the feet. The magnetic clamping allows the robot to complete the full inspection task inside a submarine tank, yet without the electromagnetic feet the platform would still able to thoroughly verify the functionality of the algorithms, which is the main focus of the collaboration. Developing electromagnetic feet for the robot is the subject of future work and will not be the focus of this thesis.

1.3 Processes for an Autonomous Inspection Platform

Software based on algorithms that enable adaptation of actions according to input data is the core of autonomy. With a suitable robotic platform in mind, the remaining challenge was to implement a set of algorithms that would allow a multi-legged robotic platform to perform an autonomous inspection. To enable the robotic platform to perform an autonomous inspection, it requires the fundamental processes that allow it to act independently. These processes are;

1) *a mapping system,*
2) *an inspection planner,* and
3) *a motion planner.*

These three concurrent processes rely upon each other and require an appropriate processing architecture, herein called the *planning architecture,* available on the robot platform. This is discussed in Section 1.4. The development of each of these three processes were the focus of the collaboration.

1.3.1 Mapping System

The *mapping system* provides the robot with the ability to perceive its environment. The algorithms that underlie the sensing process utilise the onboard sensory equipment, such as a lidar, to create a model of the environment. From this model, a situational understanding of the environment is developed.

A *mapping system* is an integral part of the inspection application. The information the *mapping system* generates from the acquired data effectively provides the robot a sense of vision and spatial perception that enables several underlying algorithms the ability to process what the robot sees and make informed decisions on what to do next. For an inspection application, the discovery of new or modified areas in the environment is pivotal to ensure that the subsequent processes are able to modify their plans to accommodate new changes.

1.3.2 Inspection Planner

When performing an autonomous inspection, the robot requires an inspection plan that directs the robot to use the camera to observe all the surfaces within the environment. Inspection plans can be generated by using *Coverage Path Planning* (CPP) algorithms (Choset, 2001; Galceran and Carreras, 2013). The term *coverage* is in relation to how much of the surface is observed by the camera. The output of CPP algorithms is a collision-free path that enables a robot to traverse within or around the target environment to coverage all boundary surfaces.

CPP is a derivative of *path planning*, where the objective of *path planning* is to find a path from point A to point B. CPP adds the additional constraint that the robot must move between two positions but cover all intended surfaces in-between. In this case, CPP algorithms tend to rely on *path planning* methodologies to formulate a solution. CPP algorithms also are capable of concurrently solving the *motion plan* of the autonomous platform.

1.3.3 Motion Planner

A *motion planner* is responsible for finding a collision-free path through the environment that consists of a sequence of actions that enable the robot to physically move between each of the planned waypoints. A *motion planner* differs from a *path planner* as a *path planner* provides a path that *does not encode the mobility constraints* of the robotic platform. A *motion planner*, however, *does encode the mobility constraints* of the platform while generating a path for the robot through the environment. As such a *motion planner* can be used as a high-fidelity *path planner*.

Given the chosen platform, the *motion planner* must consider the complexities of planning a multi-legged platform. To achieve the motions required to move freely within the confined spaces of a submarine tank, each leg is required to move independently, with respect to the placement of the body and the other legs. This process requires the kinematics and kinetics to be accurately described. This process is computationally expensive, especially when replanning in response to changing conditions during execution (Short and Bandyopadhyay, 2017).

1.4 Computing Architectures to Support Inspection Planning Processes

For a robot to behave intelligently, it requires the fundamental architecture that supports such behaviour. In the previous section, the *mapping system*, *inspection planner* and *motion planner* were identified as the core components required to perform an autonomous inspection. A suitable computing architecture is required to support the three processes to allow them to execute concurrently and asynchronously with no outside intervention, that is, as an autonomous system.

There are three distinct planning architectures that can be used to enable a platform to perform autonomously there are. These are;

1) *offline*,
2) *online*, or
3) *combination* (Offline and Online).

These three options are described in the following sections.

1.4.1 Offline Planning Architecture

Traditionally, autonomous inspection plans are precomputed before execution. When planning is precomputed, it is referred to as *offline planning* as the plans are not generated during execution. A prerequisite of offline planning is that offline planning algorithms require *a priori* knowledge of the environment. Manufacturing robots such as automated *welding and pick and place machines* are suitable examples of robots following a strict procedure to ensure the job is completed as planned.

Figure 1-6 illustrates the *offline planning architecture* for an inspection planning system using the processes discussed in Section 1.3. Given a known map of the environment, an *offline planning architecture* will employ an *inspection planner* to generate a plan that will enable a robot to observe all the surfaces. To move the robot to the desired positions the

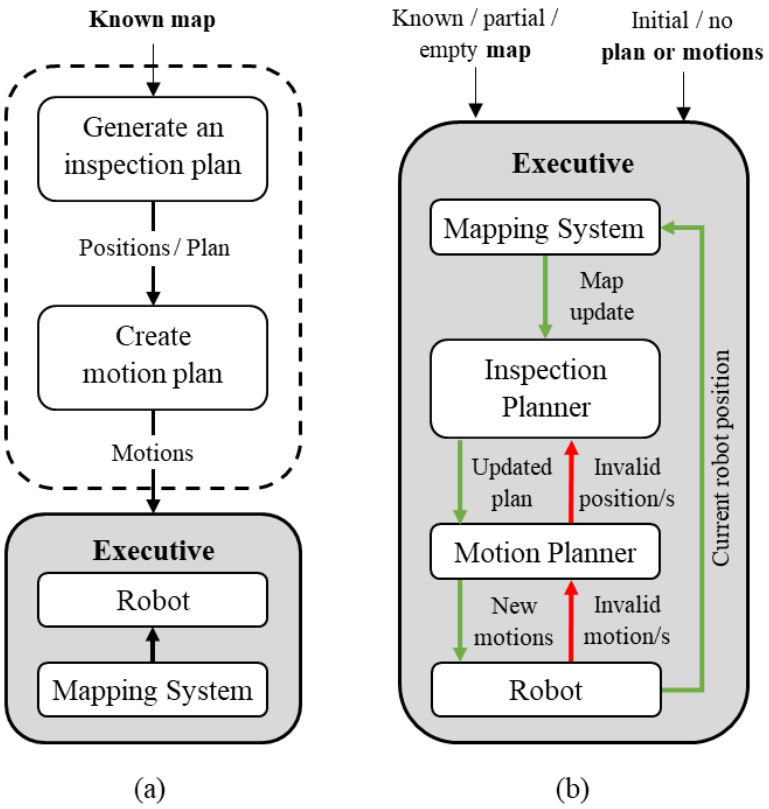


Figure 1-6: Differences between *offline* and *online planning architectures*. (a) An *offline planning architecture*. (b) An *online planning architecture*. The combination of both an *offline* and *online planning architecture* creates a *combined planning architecture* that enables an *online planning architecture* to be initialised with an understanding of the planning problem that was calculated by an *offline planning architecture* before execution. The executive is responsible for guiding the robot successfully to achieve its mission.

motion planner will compute the motions required to enable the robot to move along the calculated inspection plan. As previously mentioned, it is common for these two processes to be solved concurrently.

To execute the plan, the motion plan is provided to the robot, and under the *executive system*, will execute the pre-determined motions to allow the robot to achieve the calculated inspection plan. As the environment is assumed to be completely known to the robot, a *mapping system* is not explicitly required but can be used to detect changes that inform the *executive system* that a plan has become invalid due to unexpected changes.

The benefit of an *offline planning architecture* is that, all being well, the robot will follow a precomputed path and complete the task successfully. However, this only remains true if the environment and robot dynamic limitations are fully known and remains unchanged during execution. *Offline planning architectures* do not accommodate unforeseen changes. If changes occur within the robot's working environment, the precomputed paths that

encounter the changes would be violated, and the execution of the mission will be terminated. To overcome changes, an adaptive *online planning architecture* is required to allow for the occurrence of new information to enable the robot to continue with the prescribed mission.

1.4.2 Online Planning Architecture

An *online planning architecture* provide a solution to accommodate planning within partially known or completely unknown environments. *Online planning architecture*s implement online planning algorithms that ensure the robot has the functionality to create a plan in real-time that takes into consideration detected obstacles that will encumber the robot platform. To overcome changing situations an *online planning architecture* facilitates replanning algorithms enabling the robot to replan without the need for human intervention.

Online algorithms function to process sensory information over the lifetime of the robot mission to ensure the robot reaches the designated goal of the plan, even in the presence of an incomplete knowledge of the environment. As such, online algorithms are strongly dependant on the onboard sensing capability, such as cameras and lidar, to perceive the environment. Perception algorithms, such as *Simultaneous Localisation And Mapping* (SLAM; Cadena et al., 2016) are typically used as mapping systems to identify any new information about the environment.

Applications of online approaches can be to generate 3D renderings of an environment or building that has not been digitally rendered previously, such as archaeological surveys of ancient structures (Richardson et al., 2013) or to navigate within dynamically changing environments like autonomous vehicles (Levinson et al., 2011; Bojarski et al., 2016). The benefit employing o*nline planning architecture*s is that they provide the ability for an autonomous platform to always continue with its current mission irrespective to unforeseen changes.

Figure 1-6b illustrates how the introduction of a *mapping system* enables an *online planning architecture* to adapt the current plan to new information about the environment so compromised plans can be resolved. An *online planning architecture* provides a framework that enables the *executive* to facilitate the transfer of information between integrated mapping, inspection and *motion planner*s. The *mapping system* provides new information regarding the status of the environment. This new information enables the inspection and

*motion planner*s to adapt their plans to accommodate changes.

1.4.3 Combined Planning Architecture

In situations when an environment is assumed to be fully known before planning but it is expected that the current understanding of the environment may change during execution, it may be beneficial to initialise an *online planning architecture* with an initial plan. A *combined planning architecture* allows for the initial generation of offline plans from available knowledge of the environment, which an *online planning architecture* can refine when the plan becomes compromised during execution.

The benefits of this *combined planning architecture* are;

1) it minimises the effort required by the online planning algorithms to continually find feasible paths through an evolving environment. The cost of continually adapting plans in an evolving environment can be an expensive operation as the task completion time is non-deterministic.
2) it allows the role of an *online planning architecture* to update the existing plans on a *'need to'* basis. If the *mapping system* detects no new information throughout the lifetime of the inspection, the initial plan will execute as originally intended. However, if new information is available, the online planning approach provides the ability to correct the offline plan and allow the robot to overcome unexpected changes.

1.4.4 The Inspection Planning Framework

The implementation of the *combined planning architecture*, that combines the best functionality of both the *offline* and *online planning architectures*, is best suited to the inspection planning problem. Submarine tanks are *known* structures as they have been built to rigorous specifications. Therefore, there is sufficient information available to create an initial offline inspection plan from accurate CAD (Computer-aided design) data.

While the main structural components, such as the I-beams, are expected to be where initially designed, changes are expected to occur that would invalidate the offline plan. Changes could occur due to either;

1) potential damage to the tanks sustained during service, or
2) incremental changes in the design over the manufacturing of these vessels, ancillary components may not be where they are indicated on the CAD models.

Regardless of the cause of the changes, if an autonomous robot is to perform a successful inspection, it is critical that these changes are captured and reflected in the final 3D renderings.

Selecting a *combined planning architecture* as a suitable architecture to accommodate the three processes. This architecture created the *Inspection Planning Framework* (IPF) that will be deployed on the concept demonstrator (Figure 1-6). It will be expected that an offline inspection plan will be generated from a known model of the environment and then supplied to the *online planning architecture*, programmed as the IPF on the robotic platform, to actively update the plan when required, to respond to changes within the environment. The IPF was implemented within the Robotic Operating System (ROS; Quigley et al., 2009).

1.4.5 The Primary Role of Each Process in the IPF

Given the three processes discussed in Section 1.3 are essential for an autonomous inspection, this section explains the roles of the *mapping system*, the *inspection planner* and the *motion planner* within the IPF. Each process will be used collectively to adapt an existing inspection plan to changing conditions during execution.

Mapping System Module

The primary role of the *mapping system module* is to correlate the scans of the lidar and associate them with the known model of the environment to confirm that the perceived environment matches the known representation. For all the derived surfaces that do not correlate to the known model, the *mapping system module* will trigger a replanning update and inform the *inspection planning module* of the new surfaces that require observation. While the *mapping system* updates the representation of the environment, the surfaces are not deemed to be covered until a high-quality inspection of the surfaces have been taken by the onboard cameras.

Within the IPF, the *mapping system module* is also responsible for informing the *executive system* of gaps in map. It is expected that the detection of new features will not be immediately complete. The detection of features will be partial in nature and therefore will create gaps until they are uncovered. If the *mapping system* determines that gaps exist within the map that are not going to be detected along the existing inspection plan, the *mapping system module* can inform the *executive system* to interrupt the current inspection plan to direct the robot to new locations to uncover these surfaces. This process is commonly referred to as *exploration planning* and is combined within the *mapping system module* to

ensure all surfaces are detected in the environment. Upon detecting the new surfaces that fill the gaps, the *inspection planner module* can adapt the existing inspection plan to cover these new regions.

Inspection Planning Module

The *inspection planning module* is the principle path planning process used to direct the robot around the environment to acquire high-quality observations of the surfaces using the onboard cameras. To create the initial inspection plan, the *inspection planning module* will implement an *offline coverage planner*. Upon receiving an update from the *mapping system module* that changes have been detected, the *inspection planning module* will revert to using the *adaptive coverage planner* to assess the changes and adapt the current path to provide new viewing locations to observe these changes.

In the IPF, the *inspection planning module* will call upon the *motion planning module* with a start and finish position to resolve motion planning queries for a multi-legged robot. The *motion planning module* is required to provide a sequence of actions to create a path between two waypoints but is not required to maintain coverage of surfaces when generating the path. It is the role of the *adaptive coverage planner* to ensure coverage is maintained irrespective of the generated motion plan.

Motion Planning Module

The role of the *motion planning module* is to instruct the robot to move to the next desired location. It resolves any motion planning queries that have been requested by the *inspection planning module*. Like the *offline coverage planner*, the *offline motion planner* is responsible for providing motion plans for a multi-legged robotic platform to generate the initial offline inspection plan. When planning online, an *adaptive motion planner* is required to overcome changes in the environment.

In the IPF, the *motion planning module* has the ability to communicate with the *inspection planning module* to inform the *inspection planning module* if a viewing location was achieved. If the *motion planning module* determines that a viewing location is not reached, the *inspection planning module* can then provide additional positions until the *motion planning module* is satisfied.

1.5 Thesis Problem, Scope, Outline and Contributions

Of the three processes that comprise IPF, this thesis focuses on the development of the *adaptive coverage planner* to enable the *inspection planner module* to replan an inspection plan for an autonomous platform performing an inspection. In the following sections, the requirements associated with the *inspection planning module* are discussed to ensure the *adaptive coverage planner* works effectively within the IPF. With these requirements in place, the scope, original contributions and outline of this thesis are presented.

1.5.1 Submarine Tank Inspection Planning Problem

For the *inspection planning module* to be a successful component of the IPF, a list of requirements was placed on the development of the *adaptive coverage planner*. These requirements define the *submarine tank inspection planning problem* (STIPP) criteria.

Requirement 1: Construct an inspection plan from a known 3D model of the environment.

Requirement 2: The inspection plan should consist of discrete viewing positions that enable the robot to stop and photograph desired surfaces.

Requirement 3: The *inspection planning module* should implement a coverage planner that has the ability to generate an inspection plan for a variety of different tank variations without parameterisation of the environment.

Requirement 4: The *inspection planning module* should assure either complete coverage or the highest attainable coverage of the interior tank surfaces, including all internal fittings and reinforcement structures.

Requirement 5: The *inspection planning module* should generate coverage plans that accommodate a multi-legged, high-DOF robotic platform.

Requirement 6: The *inspection planning module* should contain an internal framework that allows adapting an existing offline inspection plan for coverage of newly detected features.

Requirement 7: The *inspection planning module* should provide inspection plan updates in a timely manner.

As previously discussed, the design and layout of the constructed submarine tanks are known. Therefore, providing an offline inspection plan using the techniques of CPP algorithms should be sufficient to supply the IPF with an initial inspection plan (*Requirement 1*).

Given the ideal robotic platform possesses holonomic mobility constraints and has the ability to stabilise it at any given location to take photographs, the assigned viewing locations generated by an *inspection planning module* are required to be discrete in nature (*Requirement 2*). The collection of discrete viewing locations will aid the creation of a 3D rendered model for external examination. This requirement enables the inspections to be repeatable and allows the robot to autonomously navigate back to a known location if further investigation is required.

It is imperative that when generating these viewing positions for the robot the positions are not generated through any direct parameterisation of the tanks (*Requirement 3*). Issues pertaining to the exact parameters of the tank dimensions is defence classified information and not available for public release. Furthermore, direct parameterisation would be infeasible. The task to systematically determine the correct viewing locations is challenging due to the significant number of tanks, their variable dimensions and layouts. Therefore, the algorithms implemented within the *inspection planning module* are required to autonomously generate coverage positions based on inferred metrics rather than using direct metrics.

The *inspection planning module* should ensure complete coverage is attainable and is of sufficient quality that it can create a high-quality digital rendering of the submarine tanks (*Requirement 4*). However, complete coverage may not be achievable as the complex geometries of these spaces have the potential to invalidate several positions the robotic platform would need to acquire to obtain coverage of the surface. In this case, it is acceptable to relax the 100% coverage constraint providing a suitable attempt has been made to cover these problematic regions, and if they remain unobservable, ensure that they are duly reported.

The *inspection planning module* should be accommodating to the type of platform that is intended to be used (*Requirement 5*). As the motion planning of a multi-legged platform is an expensive process, the *inspection planning module* needs to consider the number of times the *motion planner* is used to create an inspection plan. The more calls to the *motion planner*

the more expensive planning is going to become. The approach this thesis takes is to minimise the usage of the *adaptive motion planner* when generating an inspection plan instead on relying upon the *motion planning module* to be made faster.

As discussed in Section 1.4.5, the *inspection planning module* is solely responsible for facilitating an inspection plan update. Therefore, the *inspection planning module* has to possess the ability to update an inspection plan by determining which segments of the existing plan have been completed, which sections are still remaining to be completed, and determine which segments are on longer valid due to the changes. Facilitating this update also requires the process to replan the compromised segments of the plan. These processes require an internal framework that can accommodate the algorithms required to perform an online inspection plan update (*Requirement 6*).

Of all the requirements, producing an updated plan in a timely manner is the most subjective (*Requirement 7).* As there has been no reports on such a platform, such as the PhantomX, performing autonomous inspections within submarine tanks, it is unknown how long an inspection plan will take to compute or execute. Without a guide or an indication of a suitable time to produce a plan, or even execute the plan in an online context, it might be suitable to solve the offline inspection plan first and then determine how it should behave online.

It is assumed that the *guidance system* encapsulating the IPF will concurrently replan the inspection plan as the robot is performing the inspection. The robot sensing range is significantly greater than the body of the robot. Therefore, changes can be detected well before the robot can get there and given that this robot will not move quickly as it moves through the complex spaces, replanning could be achieved before it arrives at the change.

A safe logical argument would be to assume that replanning times of over an hour would not be suitable. Planning times in the tens of minutes would also not be desirable because if many changes were to occur, several hours could be spent with the robot waiting for planning updates. Therefore, an appropriate interval would be in the minutes, preferably in less than two minutes. One could safely assume that given the slow traversable speed of the platform, a one to two minute replan times is ample replanning time while the robot is concurrently performing an inspection.

1.5.2 Thesis Scope

Given the STIPP criteria, the scope of this thesis includes the development of a *3D adaptive coverage planner* that enables an autonomous robot to perform a comprehensive inspection within confined spaces. Specifically, the confined spaces of submarine tanks. The algorithm includes the ability to replan in real-time when new information that differs from the a priori environment presents itself during execution. This thesis will focus predominantly on the development and analysis of an *adaptive coverage planner* and its associated algorithms to ensure it can provide timely updates to changing conditions.

This thesis does not develop a *mapping system* or provide a solution to solve the *adaptive motion planning problem* for a multi-legged, high-DOF robot. The development of these processes is the focus of other colleagues in the project group. Details summarising each of these processes can be found in Pivetta et al. (2017) and Short and Bandyopadhyay (2017).

At the time of completing this thesis, neither the concept demonstrator nor the *adaptive motion planner* was available for real-world testing. Therefore, this thesis focuses primarily on the development of an *adaptive coverage planner* by thoroughly analysing the intrinsic behaviour of a designed coverage planner via simulated trials. Furthermore, as access to the real-world submarine tank data is strictly classified information, this thesis alternatively presents simulations over a synthetically designed submarine tank model that contains features akin to what would be expected in the real-world counterparts.

To ensure the *adaptive coverage planner* was not solely dependent upon a particular robotic platform or the two other IPF modules, it was developed in isolation. To ensure that when real-world testing was to occur, integration could occur simply, assumptions that reflect the expected interactions between the *mapping system*, *motion planner* and robotic platform, were placed on the *adaptive coverage planner*. These assumptions ensured that the STIPP criteria would still be satisfied but the results generated by the simulations would not be limited to only these properties. Therefore, the findings of this thesis can be applicable to wider variety of robotic platforms and planning scenarios. In short, this thesis made the following assumptions about the relationships between these systems:

Assumption 1: The *adaptive coverage planner* integrates with a *mapping system* that provides mapping updates only when new features are detected in the environment.

The *mapping system* does not detect features that have been moved within, or been removed from, the environment. As *Requirement 4* of the STIPP specifies, the *adaptive coverage planner* has to ensure complete coverage of these new surfaces are obtained.

Assumption 2: The *adaptive coverage planner* is not responsible for directing the robot to seek out the uncovered regions of the environment that are created by partially constructing the environment over time of the inspection.

This responsibility was given to the *mapping system* to inform the guidance system of these information gaps so online planning algorithms can be employed to direct from the current inspection plan to detect additional information about any new structures. When the robot acquires information about these new features, the *mapping system* can provide the *adaptive coverage planner* with a map update so the current inspection plan can be updated.

Assumption 3: The *adaptive coverage planner* uses a simplified 6-DOF robot model to represent the tibia joint that contains the camera for inspection.

While it is common for path and motion planning to be coupled together and solved as one process, to minimise the expense of solving high-fidelity motion planning problems associated with a multi-legged platform, a simplified representation of the robot is used to solve the inspection plan. The *adaptive coverage planner* uses the simplified representation to provide the high-fidelity *motion planner* with an approximated plan that is required to be solved explicitly. Regardless of the actual paths the *motion planner* solves, the approximate solution given by the *adaptive coverage planner* still maintains complete coverage.

As the inspection plan approximates the actual plan, it is evidently not going to be an optimal solution. Given the complexity of moving an autonomous robot in a complex environment, such as a submarine ballast tank, *this thesis is not concerned with the optimality of a solution but providing online updates that are feasible and achievable*. Furthermore, by choosing a 6-DOF representation, ensures that the results of this thesis will apply to robotic platforms of similar constraints and higher complexity. These assumptions are used to constrain the scope to focus on the development of an *adaptive coverage planner*. The complete list and the full details of these assumptions are discussed in *Chapter 3* of this thesis.

1.5.3 Thesis Outline

To achieve the above stated requirements of STIPP, the thesis is structured as follows:

Chapter 2 examines the path and coverage path planning literature to determine if current solutions already meets the STIPP criteria. The conclusion of this chapter provides a *gap statement* and provides recommendations to which type of CPP approach is best suited to satisfy the STIPP.

Chapter 3 selects an *offline sampling-based coverage planner* that presents itself as a suitable candidate to solve STIPP. Given that there is no online implementation of the *offline sampling-based coverage planner* that actively adapts an existing plan in partially known environments, two replanning strategies commonly used in path planning, a *full replan* and *plan repair*, are proposed to create the online variant that can satisfy all of the STIPP criteria. Potential concerns of the *offline coverage planner* that may hinder online performance are discussed. The assumptions and constraints between the relationships of the IPF modules are also discussed to constrain the focus of the thesis to the development and computational and algorithmic analysis of the *adaptive sampling-based coverage planner*.

Chapter 4 examines the *offline sampling-based coverage planner* across a series of increasingly difficult planning problems to determine if any of the listed concerns present themselves in different planning environments. An analysis of the planning data suggests sensitivity to environmental changes as well as excessive planning times of large-sized planning problems will hinder the online performance of either proposed replanning strategy. Both issues needed to be addressed to ensure online planning times are consistent and reliable.

Chapter 5 investigates the planning data from *Chapter 4* in further detail to determine the cause behind the excessive planning times exhibited in larger sized planning problems. The analysis of the planning data aids the design of *additional termination conditions* that track the behaviour of the *offline sampling-based coverage planner* and to terminate when it is unlikely to improve upon the current best-found solution.

Chapter 6 expands further upon the findings of *Chapter 4* and *5* to introduce *topological curve-skeletons* that form a new heuristic that enables the *offline sampling-based coverage planner* to factor the environmental influences. *Topological curve-skeletons* approximate the paths around obstacles to better inform the path planning process.

Given the analysis of the offline planning data conducted over *Chapters 4* to *6*, *Chapter 7* presents the first *adaptive sampling-based coverage planner* that uses a *plan repair strategy*

to provide inspection planning updates. The *plan repair strategy* uses the algorithmic properties of the *offline sampling-based coverage planner* to create *regions of interest* to segment and preserve as much of the current tour as possible to minimise the replanning effort around newly detected features in the environment.

To determine if the *plan repair strategy* is indeed the better replanning strategy, *Chapter 8* performs a computational analysis between *adaptive sampling-based coverage* executing both replanning strategies. Therefore, for completeness the *adaptive sampling-based coverage planner* implementing a *full replan strategy* is also presented. A series of simulated experiments examine the computational performance of both strategies to determine which is best suited to convert the *offline sampling-based coverage planner* for online implementation. The two *heuristics* developed in *Chapters 5* and *6* are also integrated and tested within each *adaptive coverage planner*.

Finally, *Chapter 9* amalgamates the findings on the proposed *heuristics* and experiments that compare each *adaptive coverage planner*. The limitations of the developed approaches and the assumptions that were placed on this thesis are discussed as future work.

1.5.4 Original Contributions

This thesis makes the following five original contributions to the field of coverage path planning.

Contribution 1: The thesis investigated the functionality of the *offline sampling-based coverage planner* developed by Englot and Hover (2017)[1]. The investigation determined the suitability of sampling and path planning processes for an online implementation (*Chapter 4*). The analysis of experimental data revealed that major issues were present when using large covering set sizes numbering into the thousands. These issues made it infeasible for problems of this size to be solved appropriately and thus necessitated new directions of development to improve both the offline and online planning processes (*Contributions 2 and 3*) The original contribution to knowledge is the analysis and findings of this investigation. This investigation examined the *offline sampling-based coverage planner* in a different planning domain under different planning constraints, providing further insight about the coverage planner than previously published.

[1] A republication of the 2011 paper of the same name (Englot and Hover, 2011).

Contribution 2: The investigation in *Chapter 4* revealed that the existing *offline sampling-based coverage planner* exhibited significant variability between solutions when planning over large covering sets. This thesis explored the cause behind the variability and found that it was produced by the *equality termination condition* and approximation *Travelling Salesman Problem* solutions within the *lazy point-to-point planner* (*Chapter 5*). To rectify this variability, *additional termination conditions* were developed. These new termination conditions successfully removed the majority of the planning time variability due to ineffective iterations, which consequently resulted in a significant reduction of planning times across all planning problems. These new *additional termination conditions* enabled both the *offline* and *adaptive sampling-based coverage planner* to continue using an approximation TSP solver to solve large coverage planning problems quickly.

Contribution 3: The analysis in *Chapter 4* also demonstrated the sensitivity of the *lazy point-to-point planner* to the geometry of different environments. Topological skeletons were introduced to create the *hybrid-heuristic* and increase the efficiency of the *lazy point-to-point planner* to better solve the *multi-goal planning problem* in complex spaces (*Chapter 6*). The *hybrid-heuristic* generally guided the *lazy point-to-point planner* to a faster convergence on solutions, significantly reducing planning times of large complex coverage planning problems in concave planning environments.

Contribution 4: This thesis presents a novel *adaptive sampling-based coverage planner* that is capable of performing inspections from an autonomous platform within confined, complex environments (*Chapter 7*). An adaptive variant of the *offline sampling-based coverage planner* of Englot and Hover (2017) was built by extending the planner with the capability to modify the current inspection plan to accommodate new changes within the environment (*Chapter 3*). The thesis investigated two replanning strategies, a *full replan* and a *plan repair* to extend the *offline coverage planner* into the online domain. The investigation found that the *adaptive coverage planner*, using a *plan repair strategy*, significantly reduced the computational effort required to update an existing inspection plan compared to an *adaptive coverage planner* using a *full replan strategy* (*Chapter 8*).

Contribution 5: While a *plan repair strategy* using a *region of interest* to bound the influence of change is not a novel approach to minimising the replanning effort, the application of a *plan repair strategy* to the *offline sampling-based coverage planner* has not, to the author's knowledge, been attempted previously (*Chapters 3 and 7*). Application of the *region of interest* based on the sensing capability of the visual sensor enabled the planning processes of a state-of-the-art *sampling-based coverage planner* (which has been highlighted as an expensive coverage planner and inappropriate for direct online implementation) to perform efficiently online.

1.6 Chapter Summary

The motivation behind this thesis is to deploy an autonomous robot to perform an inspection of the tanks of the Collins Class submarine to minimise the risk associated with performing these inspections manually. Given the complexities of the confined space of the submarine tanks, a multi-legged platform was selected as the preferred platform to undertake the inspection task.

To ensure the robot has the capability to perform the inspection task, three different planning architectures, *offline*, *online* and *combined* were discussed. The inspection task requires that all internal surfaces of the tank are imaged by the robot camera. As it is likely that the majority of the surfaces will correlate well with the tank design data, adopting a combined approach to planning, where an offline inspection plan is augmented by online planning methods would be a suitable solution to ensure complete coverage is maintained. The selection of the *combined planning architecture* formulated the IPF that integrates the processes of a mapping system, *inspection planner* and *motion planner* together to enable the robot to complete the inspection task.

The focus of the thesis is to develop an *adaptive coverage planner* that can adapt an existing inspection plan to changing conditions. A list of criteria, referred to as the *submarine tank inspection planning problem* was defined to highlight what a successful adaptive coverage planner would provide within the IPF. The thesis scope highlighted the assumptions placed on the thesis to constrain the focus to be on the development and analysis of the *adaptive coverage planner*. The chapter concluded with an outline of the thesis and a list of the original contributions this thesis makes to the body of knowledge. In the next chapter, the path planning and CPP literature is reviewed to find a solution to the STIPP.

Chapter 2

Coverage Path Planning for Autonomous Tank Inspection Robots: A Literature Review

2.1 Introduction

In *Chapter 1,* a list of criteria was presented that defined the parameters for the solution of the *submarine tank inspection planning problem* (STIPP). The STIPP specifies that an *adaptive coverage planner* is required to achieve full coverage of the submarine tanks. The goal of this thesis is to address the STIPP criteria by developing an *adaptive coverage planner* that can provide coverage to new and existing structures to inspect the inside of a multiple submarine tanks.

The STIPP criteria itself addresses three different by interconnected problems. The STIPP criteria has an *offline, online* and *foundational components* and therefore were categorised to address each problem. *Requirement 1* was *categorised as an offline requirement* as it requires an initial inspection plan to the constructed from a known model of the environment. *Requirements 2 to 5* were characterised as *foundational requirements* because they apply regardless of whether the coverage planner is an online or offline process. *Requirements 6* and *7* were categorised as *online requirements* as they focus on the adaptive behaviour of the coverage planner. These three categories set the direction for investigation and therefore this chapter explores offline and online path and *coverage path planning* (CPP) algorithms to determine if current techniques are applicable to satisfy the STIPP criteria. After an evaluation of the literature, a *gap statement* is presented that identifies the key areas of focus so the goal of the thesis can be achieved.

While there is a distinction between motion planning and path planning in this thesis, for the

purpose of the literature review, path and motion planning is used interchangeably. It is common for path planning for robotic systems to solve a motion plan concurrently when solving the path planning problem (LaValle, 2006). Unless specified otherwise, the output of CPP algorithms listed in this review produce collision-free paths that have considered the motion or mobility capability of the platform. However, beyond this chapter, the explicit motion planning of a multi-legged platform is handled by the *motion planning module* in the Inspection Planning Framework (IPF) and is not covered in this thesis or literature review. How motion planning is incorporated into the *adaptive coverage planner* that still manages to satisfy the STIPP criteria is discussed in further detail in *Chapter 3*.

2.2 Review of Path Planning Algorithms

Path planning or path finding, as it is termed in video gaming literature (Botea et al., 2013), is a rich and diverse field containing many methods to solve what seems to be an intuitive problem; *finding a path for a given entity through an environment from point A to B* (Figure 2-1). When it comes to path planning there is no single solution that is applicable for all applications. Some path planning algorithms focus solely on producing the *optimal* routes, or if time is an issue, other path planning algorithms produce paths in the fastest possible manner.

Choosing or designing an appropriate path planning algorithm is dependent upon:

1) The application of the path planning algorithm to find the optimal path, or a feasible path given the time, or both.
2) The motion planning constraints placed upon the platform.
3) The environment in which the planning is undertaken.

Of the three listed dependencies, how the environment is represented plays an important role in determining which planning algorithms are applicable to solve the planning problem (Buniyamin et al., 2011). Finding a suitable representation of the environment that can be processed by a computer system requires discretisation. As such, paths solved using a discretised environment will only ever approximate the true path in $\mathbb{R}^2$ and $\mathbb{R}^3$.

Common approaches in path planning to represent the environment are:

1) To decompose the environment's traversable space into a graph, to enable graph-based search algorithms to find a feasible path through the environment. If the information in the graph is complete before planning, the optimal answer can be found.

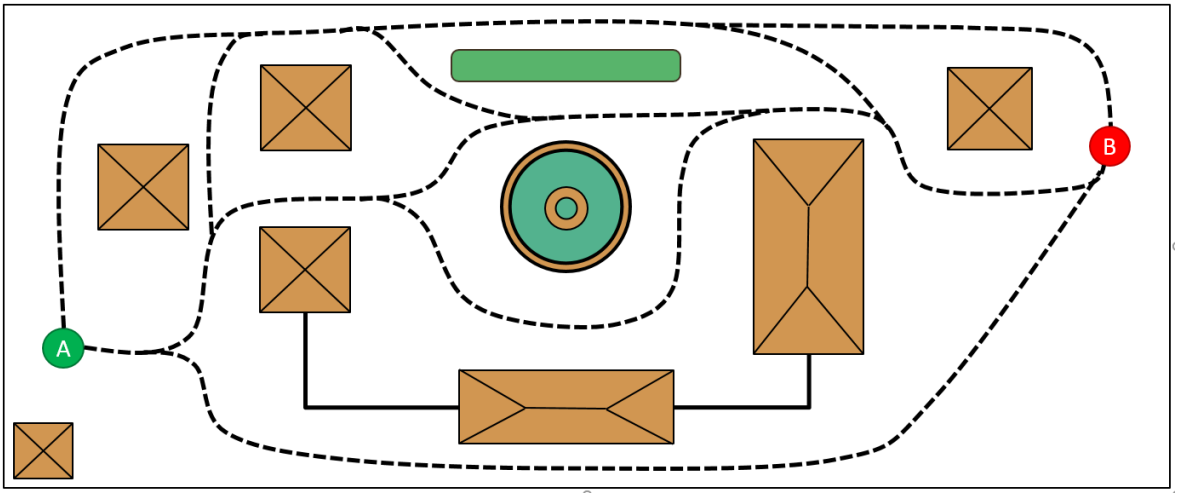


Figure 2-1: Various paths exist through a given environment. The choice of path is dependent upon the application.

2) To not decompose the environment as a graph but to enquire information about the environment via sampling. This approach is commonly used by sampling-based path planning algorithms that construct paths via sampling the space and determine the validity of a given position by performing collision checks against the environment. As the environment is sampled, paths generated by these algorithms are not optimal but generally faster than graph-based solutions.

Given that CPP is path planning under coverage constraints, this section provides an overview of the common 2D and 3D path planning approaches that generate paths using environmental abstracted graphs (Section 2.2.1) and sampling-based methods (Section 2.2.2). As the focus of this thesis is to develop an *adaptive coverage path planner* to solve STIPP, this section also includes a summary of *adaptive path planning methodologies* that actively replan the current plan within environments that contain static or dynamic entities (Section 2.2.3).

For more detail, the reader is referred to the publications of Choset et al. (2005), LaValle (2006) and Latombe (2012) that cover a wide variety of path and motion planning algorithms. These publications provide an excellent introduction to the path planning field, especially for robotic applications. For the application of path-finding techniques in video game literature, the reader is referred to Rabin (2020).

2.2.1 Graph-based Search over Decomposed Environments

A common approach to solving the path planning problem is to represent the problem as a graph. With the planning problem represented as a graph, traditional graph search algorithms

such as *Dijkstra's Algorithm* (Dijkstra, 1959) and A* (Hart, Nilsson, and Raphael, 1968) can be applied. Providing the graph is complete, both *Dijkstra's Algorithm* and A* provide optimal solutions to graph search. The difference between each algorithm is that *Dijkstra's Algorithm* searches through all nodes in the graph to find the optimal solution whilst A* requires an *admissible heuristic* to guide the search to an *optimal* solution, consequently, evaluating fewer nodes than *Dijkstra's Algorithm* in the process. Providing the heuristic is *admissible*, that is, it does not overestimate the distance to the goal, A* provides an optimal answer faster than *Dijkstra's Algorithm.*

Due to the computational advantages of A* it is considered one of the best graph-based search algorithms, popular in a number of applications outside of robotic planning such as the video gaming industry, that require path planning to be performed in real time (Cui and Shi; 2011; Botea et al., 2013; Rabin, 2015). However, for the effective use of these graph-search algorithms, a graph must be derived from the environment one wishes to plan upon.

Graphs from Grids

One of the simplest forms of creating a graph is to rasterise a 2D environment into a regular, equalled sized grid. Each grid space is termed a cell. Cells that contain no obstacles are free to traverse while those containing obstacles are not. This approach is termed an *approximate cellular decomposition* (Choset, 2001) as the grid spaces only approximate the features within the environment. As the cells are a fixed size, they only approximate the occupancy of the obstacles, therefore are likely to overestimate the size of the actual obstacle or feature.

A graph is formed by linking each cell of the grid to the neighbouring cells. Cells that contain obstacles are not included in the graph. In this form, graph search algorithms such as A* can find the path through the environment. An example of a 2D grid-based approach to path planning is represented in Figure 2-2a.

Resolution Issues

The resolution of an approximate grid is an important consideration. The resolution should be fine enough that the space is suitably represented. A course resolution will lose accuracy of the representation, while a finer cell resolution will create a larger sized graph. Significantly large graphs have the following implications:

1) More memory is required to store the graph.
2) The longer it takes for graph searching algorithms, such as A*, to find a solution.

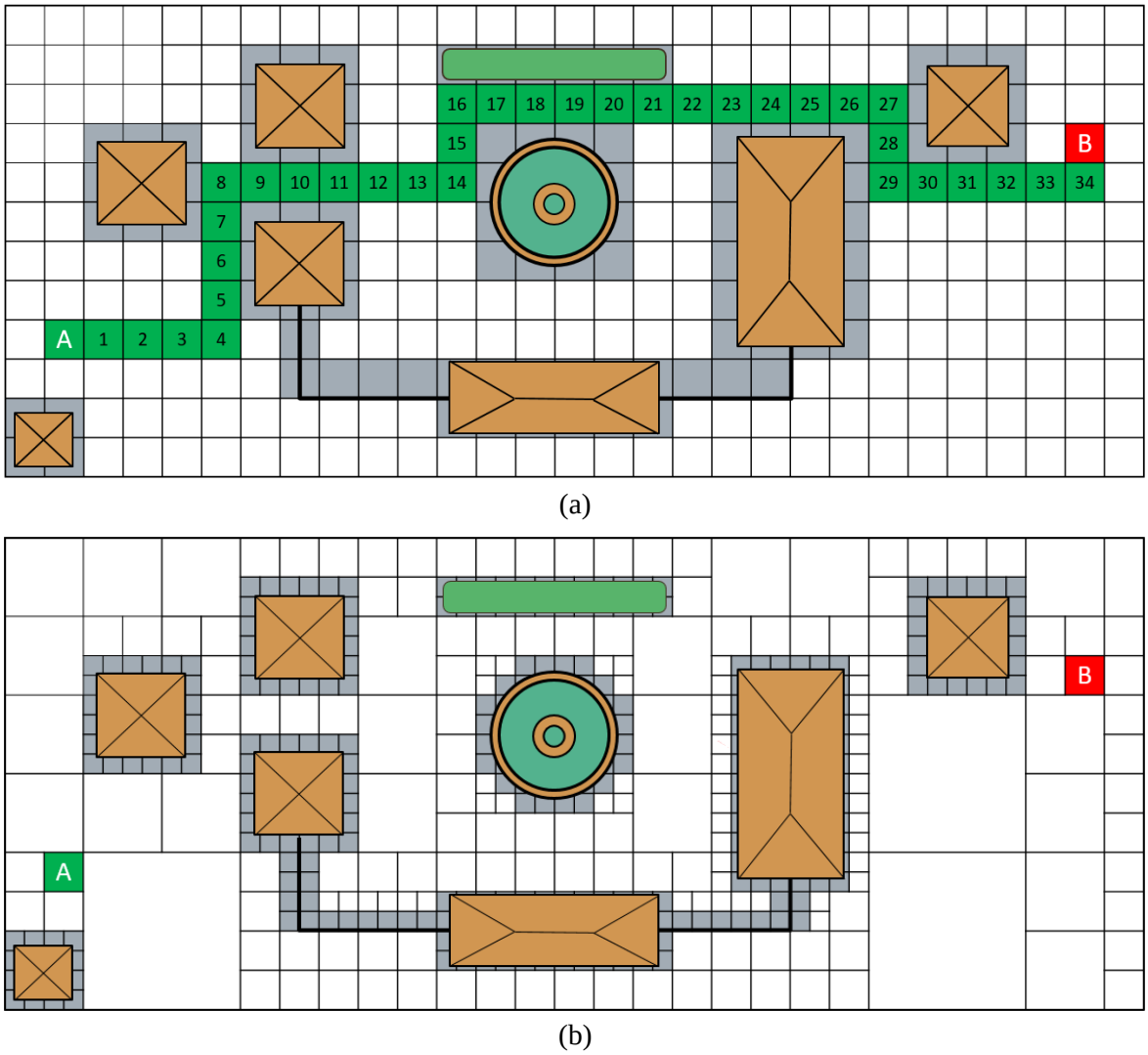


Figure 2-2: Path planning using approximate cellular representations. (a) A regular sized grid. (b) Quadtree.

Generally speaking, graph search algorithms suffer from dimensionality (Ferguson, Kalra, and Stentz, 2006). The larger either the environment or graph becomes, the more challenging it is to solve. Therefore, in principle, smaller graphs equate to faster solution times.

Quadtrees and Octrees

Quadtrees are useful for reducing the number of cells required to represent 2D environments (Samet, 1984). *Octrees* are the 3D equivalent, but cells are called voxels in 3D spaces (Meagher, 1982). *Quadtrees* decompose cells based on their occupancy. Cells that contain obstacles are decomposed down into four equally sized cells of smaller resolutions to represent the features more accurately in the environment (Figure 2-2b). Therefore, fewer cells representing the free and traversable space saves on memory and decreases search times compared to a regularly spaced grid of the same resolution. However, like most data structures, the larger the environment the more levels there are in the tree, and the more

expensive the searching becomes.

Completeness and Optimally of Grid-based Approaches

Providing the resolution of the grid appropriately represents the environment, grid-based approaches are *resolution-complete* such that if a solution exists within the graph, it will be found (Latombe, 2012). However, given that these grid representations are only approximations of the environment, the optimality of the resultant path is only optimal to the resolution of the grid (Karaman and Frizzoli, 2011). The graph search will provide the optimal answer for a graph, but the resultant path will not be *globally optimal*. A finer resolution will create a better representation therefore create an answer closer to the *global optimal*, but at the expensive of memory and computation.

Visibility Graphs

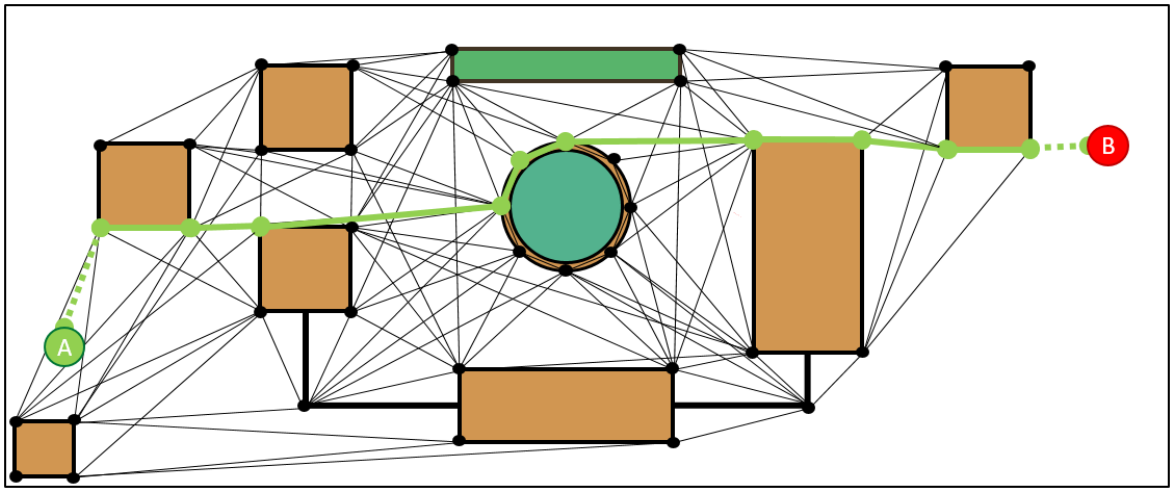
Visibility graphs have been used to create an abstract graph directly from the geometry of the environment (Lozano-Pérez and Wesley, 1979). A graph is formed by connecting line-of-sight vertices of polyhedral objects within the environment (Figure 2-3a). Finding a path through the graph is achieved by connecting the start and finish locations to the nearest vertices and solving the graph. The robot is then instructed to move along the path whilst keeping a safe distance from the objects.

Voronoi Diagrams

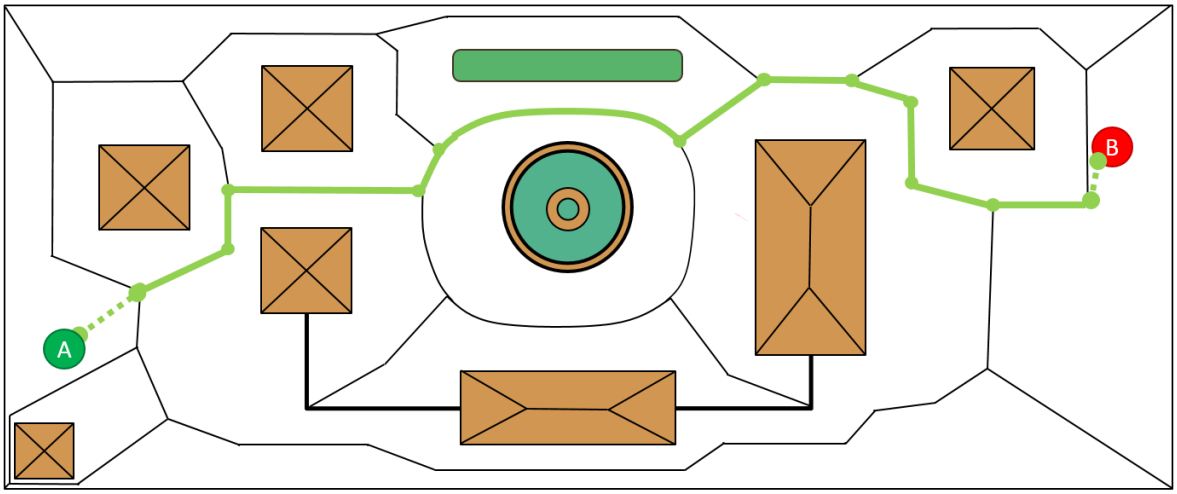
An alternative geometric presentation commonly used in 2D path planning applications is the use of *Voronoi diagrams* (Choset et al., 2005). *Voronoi diagrams* generate edges of the graph that are equidistant from every pairs of obstacles in the environment (Figure 2-3b). The ability of *Voronoi diagrams* to generate a graph with edges that create obstacle-free paths through the environment make them a popular choice for 2D robotic path planning applications as the robot can safely be instructed to move along the edges. A path through the *Voronoi diagram* can be achieved using any graph search algorithm.

Navigation Meshes

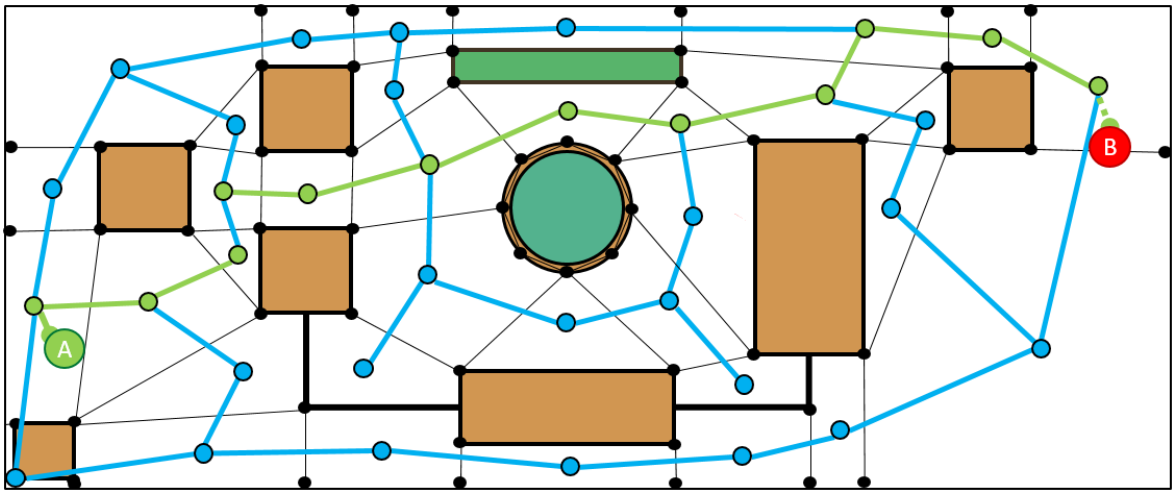
Navigation meshes (Snook, 2000) are another type of cellular decomposition method that represents the free space of complex spaces with irregular shaped cells. *Navigation meshes* are an *exact cellular decomposition* as they decompose the free space within an environment into a set of non-intersecting regions, whose union fills the target environment (Choset, 2001; Marden and Smith, 2014). *Navigation meshes* can be represented by any size and



(a)



(b)



(c)

Figure 2-3: Various illustrative examples of different forms of environmental decomposition algorithms. (a) A *visibility graph* uses the objects line-of-sight to form a geometric-based graph. (b) A *Voronoi graph* is generated equidistant between objects in the environment. (c) A *navigation mesh* made from different sized polygons. The graph of the mesh is highlighted in blue.

shaped polygon (Figure 2-3c). As a result, *navigation meshes* can represent the free traversable in fewer cells than *approximation cellular decompositions*. Fewer cells result in less memory requirements which can lead to faster path planning solutions over large environments, making them a popular choice for video game and robotic applications.

Commonly, *navigation meshes* are in the form of a triangular mesh as they are memory efficient and easier to work with in $\mathbb{R}^2$ and $\mathbb{R}^3$ (Yan et al. 2008). Creating the triangular mesh from a geometric representation of the environment can be achieved using either the *Constrained Delaunay Triangulation* (Chew, 1989) or *Delaunay Triangulation*, which is the *dual graph* of the *Voronoi diagram* (Fortune, 1995).

As cells of *navigation meshes* can represent large areas of open free space, solving the graph over adjacent cells can lead to sub-optimal paths. Planning to the centre of each cell may increase the overall travel distance while planning over the edges of the mesh can create zig-zag paths that are undesirable movements in practice. To find a better path through these cells, path planning over *navigation meshes* can be separated into two phases:

1) The first phase solves the initial path using standard graph search algorithms to find a set of traversable cells. Solving the initial path is called finding the *homotopy class* of the environment and providing a set of traversable cells is referred to as *a channel* (Kallmann, 2005; Bhattacharya, Kumar, and Likhachev, 2010).

2) The second phase applies a local planner to find a shorter route through each of the cells of the *channel*. Techniques such as the *funnel algorithm* (Lee and Preparata, 1984), *string pulling* (Johnson, 2006), *rubber-banding* (Marden and Smith, 2014), and *Dubins curves* (Dubins, 1957; LaValle, 2006) have been applied to solver shorter routes over the *homotopy class* of *navigation meshes*.

Recently, Wheare, Lammas, and Sammut (2019) demonstrated the capability of *navigation meshes* for the mission planning of autonomous surface vessels. A *Delaunay Triangulation* created a *navigation mesh* over a cluttered environment and using a *uniform cost search algorithm* (Russell and Norvig, 2016) to find *a channel*, the exact paths of the vessel were solved using *Dubins curves* (Figure 2-4).

In summary, the examples presented in this section use either a geometric representation or cellular decompositions to decompose the environment to construct a graph through free space that can be used to find either feasible or optimal paths through the environment.

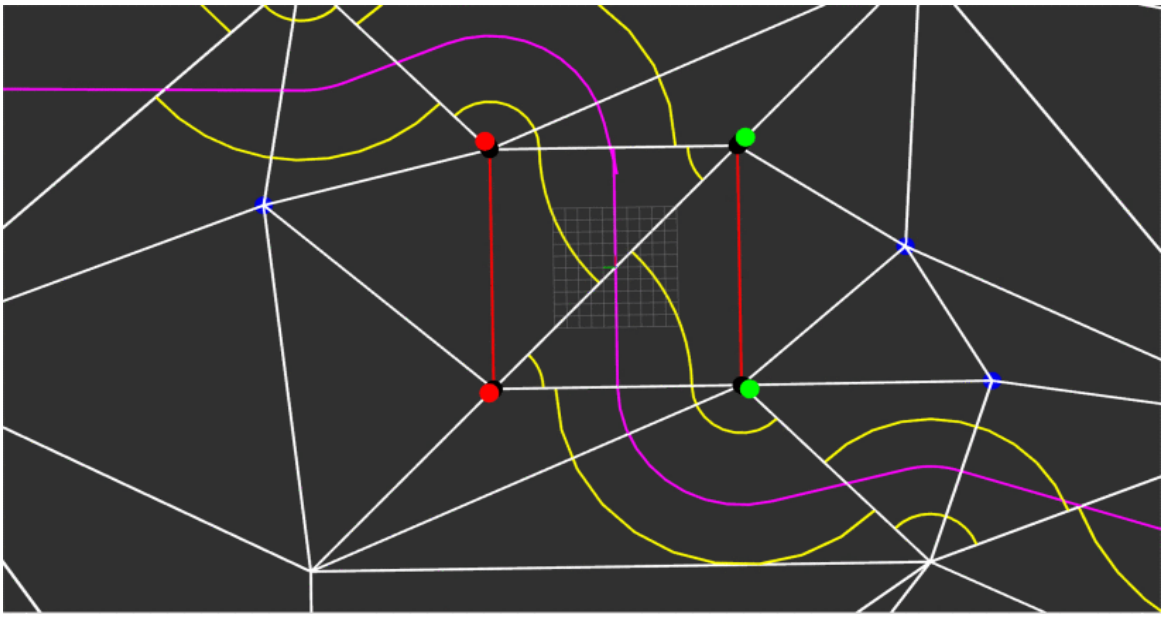


Figure 2-4: *Dubins curves* applied over a channel found over the *navigation mesh* created by the *Delaunay Triangulation* to find a path that directs the autonomous vessel through the navigational buoys (green and red) (Wheare et al., 2019; © 2019 IEEE, reproduced with permission).

Consideration needs to be given on the size of the graph and the graph search algorithm used, as larger-sized graphs become computationally infeasible to find a solution.

2.2.2 Sampling-based Path Planning

Sampling-based path planning methods do not require a searchable representation of the environment to produce a path (Karaman and Frazzoli, 2011). Path planning is conducted in the robot configuration space (LaValle, 2006), where samples that represent a robot's configuration are generally sampled at random in the space. Collision with obstacles is evaluated for every path segment, a task that is referred to as a *collision check*. When a position is sampled, the validity of the sample is checked against the environment. If valid, the position is used to construct paths through the environment. The application of using samples to seed paths through an environment, allows sampling-based path planning algorithms to work beyond the restrictive and dimensionality issues of graph-based representations.

Popular sampling-based path planning algorithms used in robotic applications are the *probabilistic roadmap* (PRM; Kavraki et al., 1996) and the *rapidly exploring random tree* (RRT; LaValle, 1998). Both planning algorithms provide fast path planning solutions in complex 2D and 3D spaces and in higher dimensional spaces for robotic arms and multi-legged platforms (Short et al, 2016).

Classification of Sampling-based Path Planners

Sampling-based path planners have been classified as either a *single-query* or *multi-query planner* (LaValle, 2006). A *single-query planner* resolves path planning queries between two specified locations once. These planners calculate paths when required, with no preconception of the environment. *Multi-query planners* on the other hand, precompute a topological representation of the environment, termed a *roadmap*, before resolving path planning queries. Providing the environment remains static for each planning query, the precomputed *roadmap* can be queried multiple times to resolve path planning queries between any two locations.

Sampling-based path planners can also be implemented as *anytime planners*. Generally, path planners continue to solve until a solution is complete. *Anytime planners*, which implement *anytime algorithms* (Zilberstein, 1996), provide approximations to the best solution by providing the best solution available after a specified time limit. A better solution can be obtained by providing these planners with more time. As a solution can be delivered at any time, *anytime planners* are preferable for online replanning applications, where the delivery of a solution can be critical for time-based applications.

Probabilistic Roadmaps

The PRM is a *multi-query planner* that has two distinct phases to generate a path, the *learning or construction phase* and the *query phase*. The *construction phase* constructs a roadmap of traversable paths between sampled positions while the *query phase* is used to resolve path planning queries. In the *learning phase*, samples are drawn at random across the configuration space to build a *roadmap* of valid configurations achievable by the robot. Samples that collide with objects are rejected. Those samples that are within free space, are connected to neighbouring samples by a local planner, which generally connects samples by a straight line but can be used to resolve paths with platform constraints of further complexity, to build a roadmap of paths (Figure 2-5a). The *construction phase* concludes once a specified number of samples have been drawn.

To resolve a path planning query using the PRM, any pair of start and finish locations can be added to the roadmap and the shortest path can be solved using as *Dijkstra's Algorithm*. Providing the environment remains static, the PRM can be continually queried to solve path planning problems.

Rapidly-exploring Random Trees

As mentioned, the RRT is another popular choice to solve path planning problems in $\mathbb{R}^2$ and $\mathbb{R}^3$. The *single-query planner*, constructs a path by spawning a tree from the start position and iteratively adds random sampling configurations to the tree as it grows towards the goal (Figure 2-5b). Like the PRM, each sample drawn is checked for collision and removed if invalid. Valid samples are connected to the nearest neighbours of the tree. The tree continues to grow towards the goal and terminates with the only path it has found. Due to simplicity of sampling free space, the expansion of the RRT occurs quite quickly and hence its popularity to solve high-dimensional path planning problems.

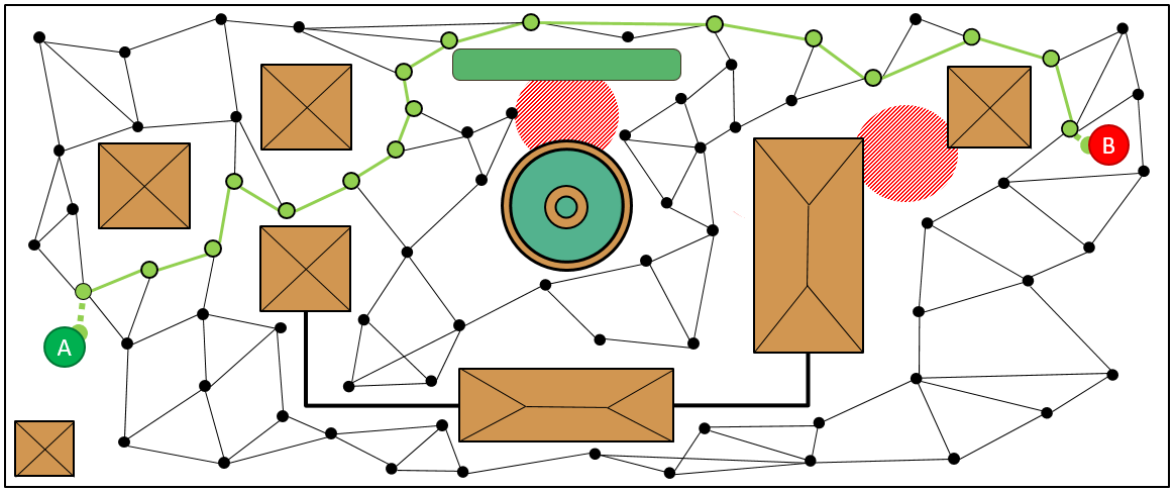
LaValle and Kuffner (2001) extend the capabilities of the RRT to derive trees that can accommodate robotic platforms with different mobility constraints. Furthermore, to increase the convergence of the RRT, the *RRT-Connect algorithm* was developed by Kuffner and LaValle (2000). RRT-Connect spawns two trees from the start and finish locations and biases the sampling of the two trees towards each other and subsequently, once near enough, connect them together. Results showed that the *RRT-Connect algorithm* improved upon solution times in environments containing more open regions compared to the traditional RRT.

Completeness and Optimality in Sampling-based Approaches

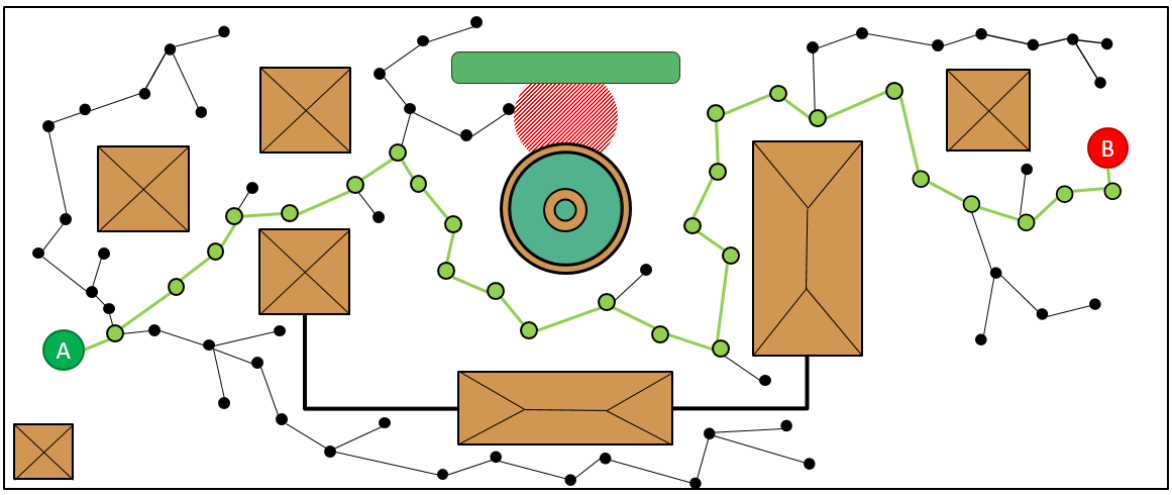
Similar to the *resolution completeness* of cellular approaches, sampling-based approaches, such as the RRT and PRM, have been found to be *probabilistically complete* (Kavraki, Kolountzakis, and Latombe, 1998; LaValle and Kuffer, 2001). If a solution can be found, the probability of finding an optimal solution converges to one as the number of samples approach infinity. However, despite the RRT and PRM notions of *probabilistic completeness*, these sampling-based path planners do not guarantee optimality.

The uniform random sampling procedures make it unlikely an optimal solution can be achieved as sampling does not explicitly capture the connectivity of the environment (Elbanhawi and Simic, 2014). Sampling in narrow areas makes it difficult to ensure that the environment is suitably covered. Figure 2-5 provides examples to how narrow spaces can be missed but how both the PRM and RRT still find feasible solutions.

Various alternative sampling schemes have been proposed to provide better coverage of the environment (Elbanhawi and Simic, 2014). Rodriguez et al. (2008) proposed a sampling



(a)



(b)

Figure 2-5: Examples of sampling-based path planning algorithms, (a) a *probabilistic roadmap* (PRM) and (b) a *rapid-exploring random tree* (RRT). The shaded red areas highlight narrow regions that samples were not generated within. Sampling within these regions may have resulted in shorter paths to the goal.

method that factors the *entropy* of the environment, to increase the sampling in narrower regions and less in open areas. However, as Elbanhawi and Simic, (2014) concluded, the effectiveness of different sampling procedures is still an open field of research and no proposed sampling method stands out above all others for every planning situation.

Asymptotically Optimal Sampling-based Path Planners

To improve upon the optimality of the paths produced by both PRM and RRT, Karaman and Frazzoli (2011) presented PRM* and RRT*. These *incremental improvement algorithms* first solve the plan quickly using their respective method and then incrementally add more samples around generated solutions to optimise the quality of the path. Given their respective improvements procedures, PRM* and RRT* were proven to be *asymptotically optimal* while still retaining the properties of *probabilistic completeness* (Karaman and Frazzoli, 2011).

The incremental nature of the RRT* solution enabled the *single-query planner* to function as an *anytime algorithm*, such that the RRT* continues to optimise the solution to a specified time limit (Karaman et al., 2011). Given the properties of RRT*, it has become a popular choice for path planning in real-time applications. The popularity of RRT* has seen several extensions to decrease the convergence time of RRT* solutions. Solutions such as;

1) RRT*-Smart (Islam et al., 2012; Noreen, Khan, and Habib, 2016),
2) Informed RRT* (Gammell, Srinivasa and Barfoot, 2014),
3) RRT$^{\#}$ (Arslan and Tsiotras, 2013), and
4) RRT^{X} (Otte and Frazzoli, [2015; 2016]).

These are just a few of the available variants that increase the efficiency of RRT* in static and dynamic environments for robot platforms that contain either holonomic or non-holonomic mobility constraints.

RRT* has also been coupled with graph-based path finding techniques in a hierarchical planning approach to find the *homotopy class* of paths through an environment (Brunner, Brüggemann, and Schulz, 2013). The initial path was first solved over grid-based representation using a graph-based search algorithm to limit the RRT* to find a higher quality path through the *channel* to the goal position.

Due to the effectiveness and computational efficiencies of sampling-based path planning to solve paths quickly in high-dimensional spaces, they continue to remain at the forefront of path planning research and development. For further information on sampling-based path planners, the reader is referred the comprehensive review by Elbanhawi and Simic, (2014).

2.2.3 Path Replanning Strategies

When a precomputed plan becomes compromised during execution, a replanning strategy is required to actively revise and modify the current plan to changing conditions. Conditions for changing an existing plan could be due to;

1) a change in the mission objective,
2) inadequate energy or resources to complete the mission, or
3) segments of the plan become no longer traversable due to the detection of either static or dynamic (moving) objects that impeded movement.

If any of these events occur during the execution of a plan, the original plan needs to be revised to ensure the primary or new goal of the mission is still maintained.

When formulating a replanning strategy for an unmanned aerial vehicle (UAV), Wzorek, Kvarnström, and Doherty, (2010) discussed three common replanning strategies used to rectify a compromised plan to changing conditions. The three replanning strategies proposed were to perform either;

1) *full replan*,
2) *partial replan*, or
3) *plan repair*.

The difference between these strategies is how much of the original plan is intended to be replanned. Figure 2-6 provides an illustrative demonstration of the three replanning strategies. A *full replan strategy* discards the entire plan from either the current location or the next immediate waypoint in the plan to make way for a new plan to the goal position (Figure 2-6b). A *partial replan strategy* conserves the replanning effort to the immediate position preceding the obstruction to the goal position (Figure 2-6c). All path segments preceding the last valid waypoint are preserved while the compromised path from the obstruction is replaced with a new plan to the final goal. Finally, a *plan repair strategy* only seeks to repair the segments of the plan that are actually impacted by the change in the environment that triggered the replan (Figure 2-6d). A *plan repair strategy* preserves all segments of the plan that are not impacted by the change, therefore minimises the replanning effort just around the compromised region.

Of the three approaches, the *plan repair strategy* presents as the quickest, as the replanning effort is focused only around the change as it attempts to minimise the amount of replanning required. However, the computational efficiency of the *plan repair strategy* decreases the more replanning of the existing path that is required. While the *partial replan* and *plan repair strategies* do present as the faster option to replanning than the *full replan strategy*, they do come at the potential loss of path optimality, as these strategies resolve the path replanning problem locally instead of globally (Figure 2-6). The trade-off between these methods is problem specific. Path planning problems that require immediate resolution would opt to perform a *plan repair* strategy to overcome the immediate changes that compromise the existing plan (Bertola and Gonzalez, 2013; Wzorek et al., 2010). However, if time is not an issue but possibly the energy consumption of the robot is of concern, a *full replan strategy* may be more appropriate to ensure the new plan is optimised to the requirements of the robot.

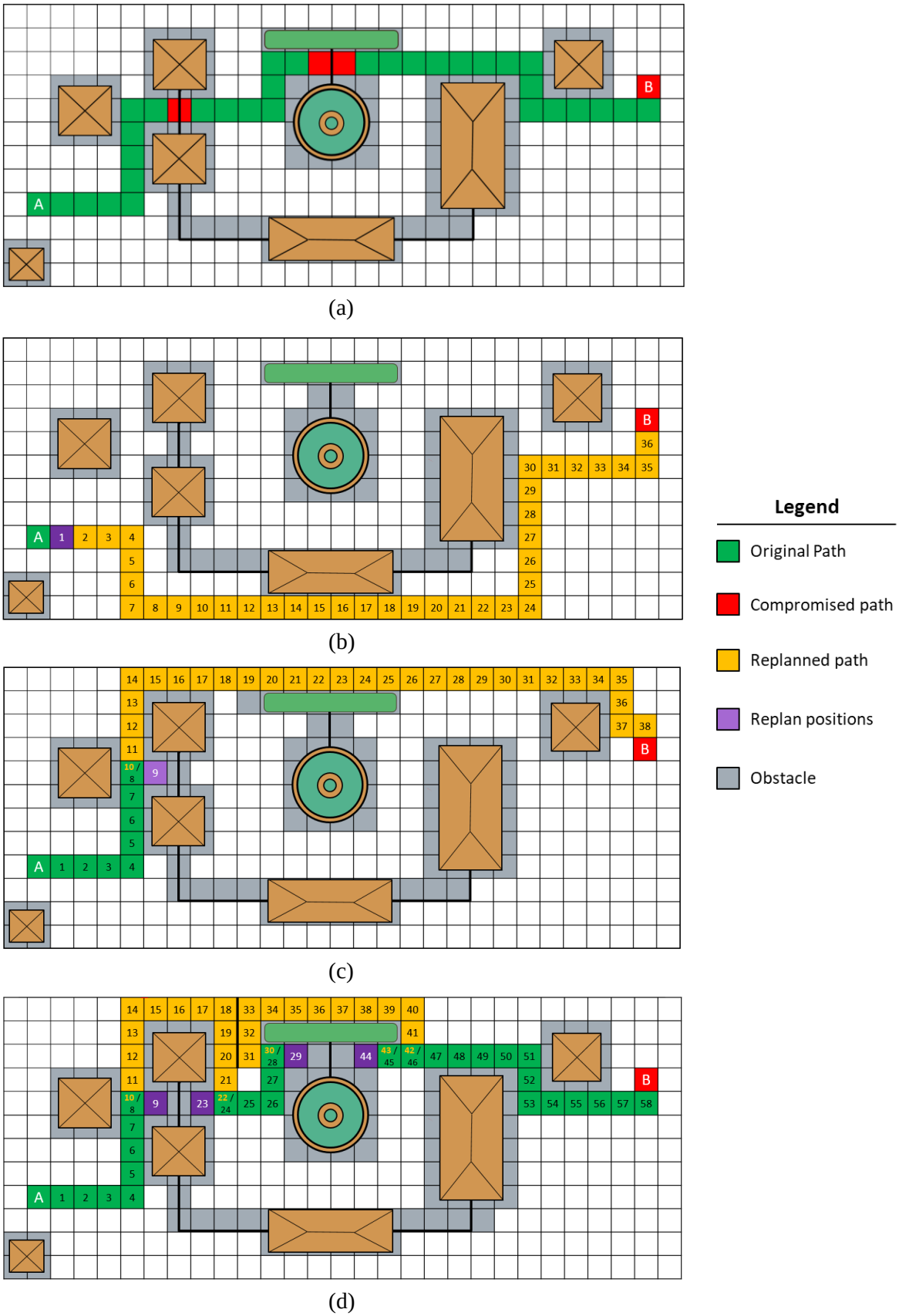


Figure 2-6: Examples of the three replanning strategies. (a) The new changes obstruct the existing path. (b) A *full replan strategy* replans from the point of detecting an obstacle to the goal position. (c) A *partial replan strategy* creates a new plan from the obstruction to the goal. (d) A *plan repair strategy* only resolves segments of the tour that have been impacted by change.

Replanning in Graph-based Approaches

For graph-based approaches, replanning can be achieved by rerunning A* when new events occur that impact the graph. Weights on the edges of the graph, which usually represent the distance to neighbouring nodes or the goal, can be updated to reflect the new change and solved again. However, continually discarding the current plan for a *full replan* can be expensive, especially when replanning over larger environments (Stentz, 1997). Therefore, when planning within partially or completely unknown environments, where information about the environment is expected to continuously evolve, it may be more appropriate to refine a solution rather than continuously calculating a new solution from scratch.

HPA* (*Hierarchical Path-finding A**; Botea, Muller, and Schaeffer, 2004) and HAA* (*Hierarchical Annotated A**; Harabor and Botea 2008) are two common approaches that have been applied to grid-based representations that incrementally use A* to find paths through both large and changing environments. To minimise the computational effort of A*, these approaches construct an abstracted graph of the environment by segmenting the larger sized grid into smaller *clusters*. The connection between each *cluster* creates a high-level abstracted graph that represents the connectivity of the environment (Figure 2-7). An initial path is solved by first solving the abstract graph using A*. The shortest path solution through the abstracted graph identifies which *clusters* to solve the higher resolution path using A* over the grids within each identified cluster. This hierarchical approach to path planning allows several instances of A* to solve smaller representations of the environment faster than what one instance of A* can achieve over the entire space.

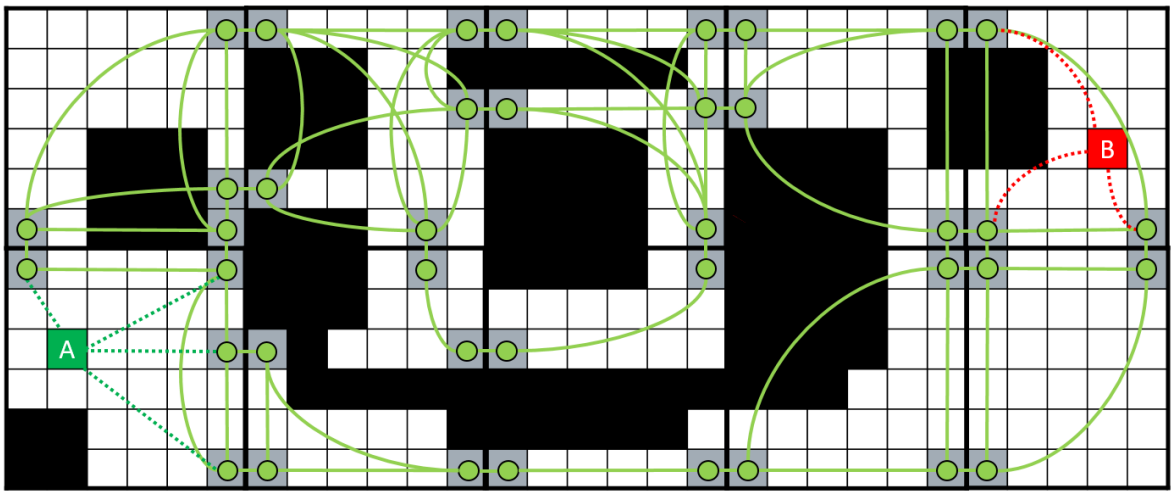


Figure 2-7: An illustration of an abstracted graph that HPA* would produce over the example planning environment. Solving a path requires first solving the abstracted graph to identify which *clusters* (6x6 cells) to solve using A*.

When changes occur online, the abstract graph can be updated and the respective paths though the clusters can be solved again. In practice, as changes are expected, to save on computational effort only the immediate clusters along the path need to be resolved. The subsequent clusters can be scheduled to be solved when required, if replanning is not required sooner.

The caveat to hierarchical approaches is because the path is solved through an abstracted representation of the environment first the resultant path is not globally optimal. This becomes the trade-off for solving large environments faster. However, path smoothing algorithms can be applied to reduce the length of the approximated paths. The equivalent approach for triangular navigation meshes is TRA* (Triangulation Reduction A*; Demyen and Buro, 2006). A more detailed discussion surrounding hierarchical approaches can be found in Vermette (2011).

To avoid repeated A* searches for each change in the environment, *incremental heuristic search algorithms* (Koenig et al., 2004), such as LPA* (*Lifelong Planning A**; Koenig, Likhachev and Furcy, 2004) and D* (*Dynamic A**; Stentz, 1997) provide an alternative way to solve the replanning problem by constructing new solutions through the reuse of information gathered from previous solutions. When a change occurs, *incremental heuristic search algorithms*, minimise the search space by only updating the weights on the edges of the graph that are relevant to finding the shortest path.

Koenig and Likhachev (2002) extend the replanning concepts of LPA* with the navigational strategy of D* to create D*-Lite. Unlike LPA*, which maintains the weight of graphs from start location to the robot's current location, D*-Lite maintains the cost from the goal location to the current location. Reserving the search direction reduced the time spent replanning towards the goal in an unknown or partially known environment. The popularity and effectiveness of the D* family of path planners has resulted in continuing advancement to improve upon the global optimality of D* paths over discrete environments in both 2D (Ferguson and Stentz, 2005) and 3D environments (Casten, Ferguson and Stentz, 2006).

Given that these algorithms are *partial replan algorithms*, there are caveats to using these methods. As these approaches attempt to preserve as much of the relevant information as possible to perform a *partial replan*, the overheads associated with preserving can become quite expensive when;

1) large changes occur that invalidate a significant portion of the current plan, or
2) changes occur quite early in the plan.

If these events do occur, *incremental heuristic search algorithms* can be less effective than a *full replan* due to these overheads associated with preserving and cross-checking existing solutions. (Koenig et al., 2004). These approaches are best to resolve small changes as the preservation of information about the planning problem enhances the ability of these algorithms to perform effectively over larger problems.

Replanning in Sampling-based Approaches

For sampling-based path planners there are also a series of applicable replanning strategies available (Short et al., 2016). Given the computational efficiencies of sampling-based path planning approaches, in particular the RRT, when planning under simple mobility constraints, in many cases it may just be easier to create a new path than it can be to repair an existing one. Automated needle steering for medical applications have sought the benefits of refining a path by continually solving multiple RRTs at each time interval and selecting the best solution that satisfies the shortest path to the goal (Patil et al., 2014). Sun, Patil, and Alterovitz, (2015) proposed *High-Frequency Replanning* where multiple RRTs were solved in parallel to provide more solutions to choose from at the each of replanning phase.

Like the *incremental heuristic search algorithms*, replanning by preservation and the reuse of previous solutions to guide the construction of a new solution is also common for sampling-based planners. The algorithms listed below are examples RRT-based algorithms that all seek to reuse previous RRT solutions, or part thereof, to refine or repair existing solution:

1) ERRT (*Execution-extended RRT*, Bruce and Veloso, 2002),
2) DRRT (*Dynamic RRT*, Ferguson, Kalra, and Stentz, 2006),
3) AD-RRT (*Anytime Dynamic RRT*, Ferguson and Stentz, 2007), and
4) MP-RRT (*Multipartite RRT*; Zucker, Kuffner, and Branicky, 2007)

As mentioned previously (Section 2.2.2), RRT* is commonly used for anytime applications with listed variants RRT$^{\#}$ and RRT^{X} being able to sufficiently replan in unknown and dynamic environments.

Recently, Hernández et al. (2019) presented results on replanning with RRT* to generate paths for an AUV moving through an unexplored environment. Their work discussed two

solutions for replanning with RRT*, other than replanning a new path each time the environment changed. The first replanning strategy replanned new paths by pruning out edges of an existing RRT* tree that can no longer be traversed due to collision. The second replanning strategy seeded a new replanning phase by utilising the *last best known RRT* solution*. The authors found that *last best-known solution* strategy resulted in a faster solution with a better success rate compared to the pruning strategy that was found to be computationally more expensive.

2.2.4 Path Planning Summary

This review highlighted common approaches to path planning in the context of coverage planning. Path planning algorithms such as *Genetic Algorithms* (Mitchell, 1998), *Potential Fields* (Khatib, 1986), *Neutral Networks* (Janglová, 2004), *Ant Colony Optimisation* (Dorigo, and Birattari, 2010) which have all been used to solve path planning problems, were not covered. These approaches are not particularly common in the coverage path planning field nor applicable to meet the STIPP criteria. Due to the main focus of the thesis being coverage planning and not path planning, any path planning algorithm that had not been thoroughly demonstrated in the coverage planning domain was outside the scope of the thesis. However, these approaches are interesting, and the reader is referred to the survey papers, Yang et al. (2016), Mac et al. (2016) and Patel et al. (2019) for more information about how these approaches have been applied to path planning.

In summary, when it comes to path planning and replanning, a path planning strategy should be selected based on the application. However, consideration must be given to the representation of the environment. Grid and graph-based solutions that rely heavily on graph-based solvers, such as A* are limited in dimensionality. Their applicability diminishes in 3D and over fine discretisation of the environment. Sampling-based approaches which have been shown to work well in high-dimensional spaces are limited in their ability to solve optimal paths. Incremental improvement algorithms such as PRM* and RRT* exist to counter this issue. One conclusion that can be drawn from the review of replanning strategies is that *they need to ensure the final plan is complete and achievable, so the threats that compromise the current plan are avoided within a respectable time.*

2.3 Review of Coverage Path Planning Algorithms

The first published attempt at *coverage path planning* (CPP) was developed by Cao, Huang and Hall (1988). Originally defined as *region filling*, Cao et al. (1988) applied a path planning technique that aimed to cover all the free space in a 2D environment for the purpose of autonomous lawn mowing. In developing this technique, six key criteria that govern the behaviour of a *region filling* operation were identified. These six criteria are defined as:

1) The robot must traverse and cover the whole region.
2) The robot must fill the region without overlapping paths.
3) The robot must be capable of continuous and sequential operation, without any repetition of path.
4) The robot must avoid all obstacles in a region.
5) The plan should enable simple motion trajectories (e.g., straight lines or circles).
6) An *optimal* path is desired under the available conditions.

Since defining these criteria, CPP has taken on a more general definition but the premise remains the same. Two prominent surveys by Choset (2001) and, Galceran and Carreras (2013), define CPP as, *the task of determining a path that passes over all points of an area or volume of interest while avoiding obstacles*. With advances in sensing capability, several CPP algorithms listed in these surveys utilise appropriate range and visual sensors to perform coverage of a boundary surface. Introducing sensory coverage paths to inspect free space or objects within the environment presented new ways to solving the CCP problem. Since Cao's criterion, CPP algorithms have been expanded to work over 2.5D and 3D environments in a number of robotic applications. These applications include:

1) Autonomous vacuum cleaning (Liu, Lin, and Zhu, 2008).
2) Underwater surveying of maritime structures (Englot and Hover, 2013; Galceran et al., 2015; Palomeras et al, 2019).
3) Tank and bridge inspection (Kalra, Gu, & Meng, 2006; Sehestedt et al., 2013).
4) Minesweeping (Acer et al., 2003; Dakulovic and Petrovic, 2012).
5) Agriculture and environmental mapping (Jin and Tang, 2011; Hameed, 2014).
6) Automated paint distribution (Atkar et al., [2005, 2008]; Yang et al., 2019).
7) Building surveillance and reconstruction (Cheng, Keller, and Kumar, 2008; Yu et al., 2015; Yao, Cai, and Zhu, 2019).

Like path planning algorithms, CPP algorithms are also implemented for offline and online

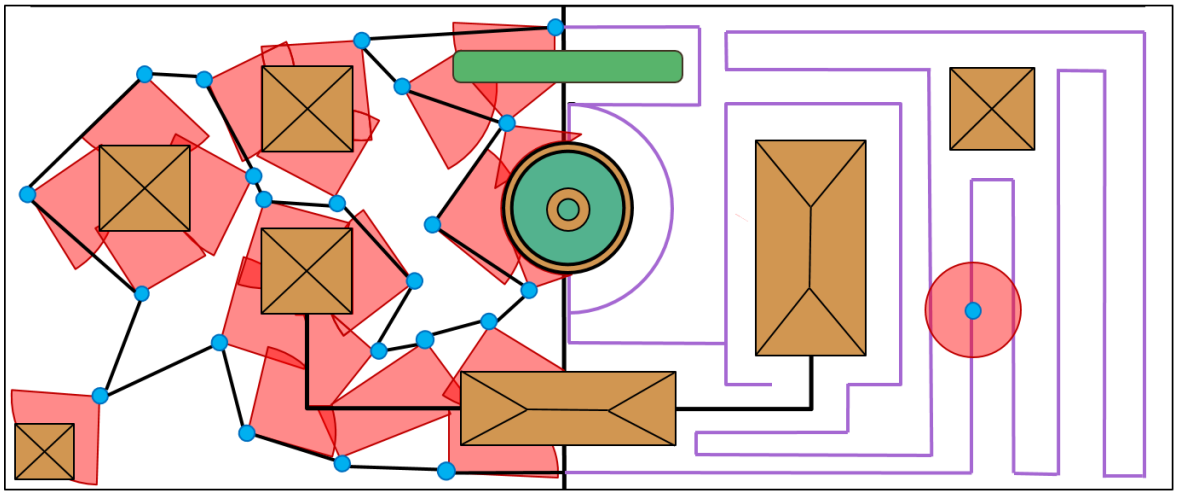
applications. As previously discussed briefly in Section 1.3.2, offline CPP algorithms solve the coverage plan before execution, while online CPP algorithms require perception to gather information about the environment to modify the current plan to account for changes within the environment. CPP algorithms can also be classified into the same two categories as path planners. These algorithms;

1) derive coverage plans from a decomposition of the environment, similar to the approaches presented in Section 2.2.1, and
2) comprise sampling-based approaches that generate coverage plans through random sampling with no explicitly searchable representation of the environment solely from the sensing capability and manoeuvring of the robot.

The main difference between these two techniques is how these coverage plans are generated. Typically, coverage planning algorithms decompose the environment into simplified regions to acquire coverage over the edges of the plan. Plans that consist of continuous trajectories enable the robot to observe all areas of the environment while the robot is moving. These methods tend to focus on traversable coverage of free space, ensuring that the robot covers all surfaces by passing within sensor range all areas of the environment.

Sampling based approaches, on the other hand, generate discrete coverage plans that typically acquire the coverage on the vertices of the plan. These plans are generally useful to stabilize a robot at specific locations to acquire the coverage. A coverage plan is created by connecting all these locations together, which collectively provides complete coverage of all the surfaces. These methods tend to focus on the coverage of the boundary surfaces of an object or environment as opposed to the free space. Given these differences, coverage planning algorithms can be further categorised as either *continuous coverage planners* or *discrete coverage planners;* a taxonomy shared in Almadhoun et al. (2016). Figure 2-8 highlights the differences between each approach.

In the literature, *inspection path planning* and *coverage path planning* have been used to differentiate between *discrete* and *continuous-based coverage path planning* respectively. However, this does not mean that all *inspection path planners* acquire coverage on the edges of the plan. Whilst the construction of the plan is discrete in nature, the paths formed by sampling can be used to create plans that observe or cover the surface while the robot is moving. This depends upon the type of inspection that is performed, and the sensors used to observe the environment. As such, if coverage of a surface is performed using a lidar, there



Coverage on the *vertices* of a plan Coverage on the *edges* of a plan

Figure 2-8: Example between coverage acquired over the edges of a plan by sampling vs coverage acquired over the edges of a plan with continuous sweeping trajectories. The red fans indicate the coverage of the surface by the robot (blue).

would be no need to remain static to acquire coverage during the inspection. Conversely, if coverage is to be taken with a camera, it may be desirable to have the robot hold position, and therefore a set of discrete locations is preferable. While the following sections will discuss this difference in further detail, for the purposes of this review, the terms *coverage path planning* and *inspection path planning* are used interchangeably to represent a complete collision-free plan that acquires full coverage of the surfaces or environments the robot is intended to inspect.

The following sections discuss common offline coverage path planning techniques that decompose the environment into small decompositions to generate continuous coverage plans (Section 2.3.1), and those that use sampling-based approaches to generate discrete coverage plans (Section 2.3.2). Online CPP methods that actively replan in unknown and partially known environments are discussed in Section 2.3.3.

2.3.1 Coverage Planning using Environmental Decompositions

In Choset's 2001 survey, a taxonomy was presented to categorise CPP methodologies that generate continuous coverage plans to ensure complete coverage of the environment was achieved. The taxonomy included:

1. *Heuristic and randomized* approaches.
2. *Approximate cellular decomposition* approaches.
3. *Semi-approximate cellular decomposition* approaches.
4. *Exact cellular decomposition* approaches.

Heuristic and Randomized Approaches

Heuristic and *randomised approaches* use predefined robot motions to randomly cover the target environment. As these approaches are not deterministic, *heuristic* and *randomised approaches* are not as effective as complete coverage algorithms, such as *cellular decompositions*, that decompose the target environment to create a coverage plan (Acar at al. 2003).

Approximate and Semi-approximate Cellular Decomposition Approaches

As discussed previously (Section 2.2.1), *approximate cellular decompositions* decompose the environment into a set of equally sized cells (Choset, 2001). For coverage-based problems, the resolution of the cells is generally based upon a footprint of the robot and coverage is achieved by the robot passing through all the cells. If the robot or an obstacle occupies just a portion of a cell, it is considered fully occupied and hence the approximation of the decomposition does not ensure a full cover of the environment. The *Wavefront algorithm* (Zelinsky et al., 1993) and the *Spanning-Tree Covering algorithm* (Gabriely and Rimon, 2002) have been common approaches to solve CPP problems over *approximate cellular decompositions*. In contrast, *semi-approximate cellular decompositions* decompose the environments to cells of a fixed width but no restriction on height (Choset, 2001). Coverage within each cell is achieved by using zig-zag motions along the grid lines.

Exact Cellular Decomposition Approaches

The most recognised and accepted method to decompose an environment into cells is to use *exact cellular decomposition* approaches. A well-recognised *exact cellular decomposition* is the *boustrophedon decomposition* (Choset and Pignon, 1998; Choset, 2000). This decomposition was developed to reduce the number of cells and travel redundancy produced by the *trapezoidal decomposition* (Latombe, 2012). Coverage of each cell is achieved by generating simplistic back-and-forth motions over each cell. Then by solving the *travelling salesman problem* (TSP; Applegate et al. 2011) between cells, the resultant coverage plan can be generated.

Exact cells in the *boustrophedon decomposition* are created by identifying the *critical points* within the environment (Choset and Pignon, 1998). Choset et al. (2000) extended the *boustrophedon decomposition* to detect *critical points* using *Morse functions* (Milnor, 2016). Using *Morse functions* coverage planning could be performed in environments that contained non-polygonal objects. Acer et al. (2002) highlights the effectiveness of *Morse*

decompositions, an *exact cellular decomposition* determined by *Morse functions*, to solve 2D and 3D coverage-based applications.

Cellular and Environment Decompositions for 3D Coverage Planning

In 3D, CPP shifts from the complete traversable coverage of free space to the complete coverage of surfaces and object boundaries. 3D continuous coverage planning algorithms are generally applied to robotic platforms that operate above the inspecting surfaces to acquire coverage at an offset distance from the surfaces they are inspecting. However, there have been 3D solutions that solve traversable coverage problems for agricultural applications (Jin and Tang, 2011; Hameed, 2014; Wu et al., 2019). These approaches considered environmental factors, such as elevation or terrain slopes, to better represent the planning problem to optimise travelling time and energy costs that would have otherwise been ignored or approximated in 2D approaches.

With the complexities surrounding the calculating of continuous trajectories over free-form surfaces in $\mathbb{R}^3$, several 3D continuous coverage planning algorithms decompose the environment to a representation of lower dimensions than the robot's workspace (Galceran and Carraras, 2013). Approaches by Atkar et al, (2005; 2008) and Cheng et al. (2008), decompose complex 3D structures into their geometric and topological representations to ensure continuous uniform coverage can be achieved, for the purposes of autonomous building inspection and paint distribution.

Galceran et al. (2015) developed a *continuous coverage path planner* that brings together a combination of 2D and 3D cellular decomposition approaches to provide coverage for 3D marine formations represented in the form of a 2.5D height map. Their process started by segmenting the height map into areas of low and high relief. For low relief areas, which were assumed to be flat, *Morse-based decompositions* were used to define the cellular boundaries around the high-relief areas so the *Boustrophedon decomposition* could be used for coverage over each cell. For the high relief areas, the height map was decomposed into a set of evenly spaced planar segmentations and *coverage offset loops*, similar to the work of Atkar et al. (2002). A complete coverage plan was achieved by combining the solution of the adjacency graph formed by neighboring cells of the *Morse-based decomposition* with the *coverage offset loops.*

In summary, this section has discussed CPP algorithms that generate continuous coverage plans from environmental decompositions. Typically, these techniques mainly focus on

decomposing the environment into cells, as seen previously in the path planning review, to provide uniform, traversable coverage of the surface rather than just the static inspection of a surface. For a comprehensive review of CPP methodologies that generate continuous coverage plans from environmental decompositions, the reader is referred to the survey by Galceran and Carreras (2013).

2.3.2 Sampling-based Coverage Path Planning

View Planning to Generate Coverage Locations

Sampling-based coverage path planning algorithms sample the robot configuration space to generate a discrete set of viewing locations, taking into consideration the constraints of the sensor and the robotic platform, to observe the surface boundaries of any given environment. The process of sampling viewing locations is commonly referred to as *view planning* (Scott, Roth and Rivest, 2003; Scott 2009). The *view planning problem* (VPP) is like the *art gallery problem* where the solutions determine the minimum number of *guards* required to observe all surfaces of the gallery. The difference between these approaches is that for the *art gallery problem* the guards are assumed to have an infinite sensing capability, while view planning must factor in realistic sensing constraints such as limited *field of view* and *field of depth*. However, solving this problem is known to be NP-hard and to find viable solutions requires approximations to the minimum covering set (Shermer,1992).

Solving the minimum covering set is known as solving *set cover problem* (SCP). While the SCP has also been defined as an NP-hard problem, heuristic approaches have been developed to approximate the *minimum set cover* (Grossman and Wool, 1997; Vazirani, 2013).

To solve the VPP González-Baños and Latombe (1998; 2001) proposed two different approximation methods, the first being a fully randomised approach, and the second being a *dual-sampling method*. In the case of the randomised approach, the entire environment is sampled at random until the guards can, in combination, observe all surfaces. This approach formulates a SCP which can be used to solve the *minimum covering set* of viewing locations that observe the boundary. For this approach, the authors opted to use a greedy approach to approximate the SCP.

Their second approach, the *dual-sampling method* incorporates the visibility constraints of the sensor to only sample within a neighbourhood of the boundary surface that requires

observation, instead of randomly sampling across the entire space. To do this, the *dual sampling method* first samples an unobserved primitive of the environment, before randomly sampling a set of potential viewing locations from which to observe that selected primitive. A *covering set* of the environment is formed by taking only the viewing location that *best* observed that primitive. This process continues until all primitives of the boundary have been observed. By incrementally building a *covering set*, the explicit solution of the SCP is avoided. Since the inception of the *dual-sampling method*, it has been the basis of several sampling-based coverage path planning algorithms, some of which are discussed in this review.

While these approximate methods of solving the VPP introduces overlap between viewing locations, the existence of the overlap is required to meet the goal of view planning (Scott, Roth and Rivest, 2003). For the purposes of image registration and digital reconstruction of the inspecting environment, overlap is necessary for algorithms such as RANSAC (*Random Sample Consensus*; Fischler and Bolles, 1981) to provide a solution that efficiently correlates with all the acquired images.

The above approaches only solve one aspect of generating an inspection plan using a sampling-based approach. The second phase is to connect these viewing locations together using path planning algorithms to create a single collision-free inspection plan. Methods to combine these two principles are classified as either *coupled* or *decoupled* approaches. *Decoupled approaches* separate the sampling of viewing locations and the path planning that connects these viewing locations together into two distinct steps. *Coupled approaches* combine the processes of sampling and path planning into single step. The sampling of the next viewing location is only preformed once a path between the previous viewing locations has been established. The following two sections discuss these approaches to construct sampling-based inspection plans in both 2D and 3D from fundamental view planning and sampling-based path planning techniques.

Decoupled Coverage Planning by Solving the Multi-goal Planning Problem

Compared to coupled planning approaches that solve both the sampling and path together in one step, *decoupled* sampling-based coverage path planners are required to solve an additional path planning problem that seeks to order the covering set of viewing locations to create a single collision-free plan. This problem is commonly referred to as either, the *multi-goal planning problem* (MPP; Wurll and Henrich, 2001) or, more generically the

robotic task sequencing problem (Alatartsev, Stellmacher and Ortmeier, 2015).

Solving the MPP commonly relies upon the solution from another NP-hard problem, the *Travelling Salesman Problem* (TSP; Laporte 1992; Applegate et al., 2011). A well-studied problem, the TSP seeks to find the shortest path between all cities, visiting every city once and finishing at the city it began. However, being an NP-hard problem, *approximation algorithms* and *heuristics* are used to solve sub-optimal TSP solutions (Helsgaun, 2000; Applegate et al., 2003). The *Lin-Kernighan heuristic* (Lin and Kernighan, 1973) is one of the most notable approximation TSP solvers, amongst several other approaches that solve the TSP using either *genetic algorithms* (Potvin, 1996), *minimum spanning trees* (Graham and Hell, 1985), *integer programming* (Orman and Williams, 2007), *ant colony optimisation* (Dorigo and Gambardella, 1997), or *simulated annealing* (Kirkpatrick, Gelatt and Vecchi, 1983).

Solving the TSP is only one part of solving the MPP. Formulating a graph structure to solve the TSP requires the costs between all pairs of goals to be established. This requires the paths between all the goals to be solved so a metric can be supplied to TSP solvers to solve the MPP. In open areas where connections between all viewing locations can be achieved via a straight-line connection, computation of all the paths may be achievable. However, calculating the collision-free paths between all goal locations in 3D can be infeasible due to any of the following reasons:

1) The number of goal locations could extend into the hundreds or thousands, the operation of solving all paired paths is $O(n^2)$.
2) The environment is a complex space containing many obstacles that make path planning difficult.
3) The motion planning within the constraints of the robotic platform are computationally expensive.

To solve the MPP in these spaces, a *lazy* approach can be taken (Saha and Latombe, 2003). The Saha and Latombe (2003) *lazy* approach to solving the MPP, iteratively solves the TSP over a planning problem that initially assumes all viewing locations are separated by the Euclidean distance. Upon solving the shortest path, those edges that are chosen are solved by the motion planner and the exact path lengths are updated in the adjacency matrix. The solution terminates when no further improvements can be made. The *lazy* approach solves the MPP by assuming that it is cheaper to iteratively solve the TSP than it is to calculate

paths, so the attempt to minimise the number of path evaluations was effective for solving the MPP for a high-DOF robotic arm. While this path planner is an $O(n^2)$ algorithm, the authors state failed to achieve this limit in practice (Saha and Latombe, 2003).

The *lazy* approach is just one method to solve the MPP. For alternative solutions to solve the MPP, the reader is referred to a survey by Alatartsev, Stellmacher and Ortmeier (2015). In this review, multiple variants to solving the MPP are presented which include formulating the MPP as a *generalised TSP* problem (Srivastava et al., 1969), where the MPP is solved between the clusters of goal locations. This problem is also known as the *covering travelling salesman problem* (Current and Schilling, 1989).

An early attempt of solving inspection plans in both 2D and 3D using a decoupled sampling-based coverage planning approach was proposed by Danner and Kavraki (2000). Their approached utilised the *dual-sampling method* proposed by González-Baños and Latombe (1998) and utilised path planning approaches to construct an inspection plan by connecting these locations into a single continuous collision-free route, akin to the *watchman route algorithm* (Chin and Ntafos, 1986). In 2D, the MPP was solved using a *visibility graph*. In 3D, as the *visibility graph* suffers due to dimensionality, randomly drawn samples were used to construct a PRM. Both of these solutions formed a graph that was solved using an approximation TSP solver.

Expanding upon the *dual-sampling method* and the path planning principles of Danner and Kavraki (2000), Englot and Hover (2011, 2017) developed a *sampling-based coverage planner* that used *redundant roadmaps* to inspect the complex geometries of ship hulls. The planning process retains the decoupled approach between sampling and path planning, but solves the SCP after the construction of the *redundant roadmap*. The *redundant roadmap* oversamples the boundary surfaces until each primitive is observed *n*-number of times, where *n* is the *redundancy* of the roadmap. Consequently, all valid viewing locations were added to the covering set. To solve the MPP, a *lazy point-to-point planner* (LPP) derived from the approach of Saha and Latombe (2003) was used with path planning queries being resolved using *RRT-Connect* rather than a PRM for improved performance.

While the *redundant roadmap* retains the tuneable parameter (*n*) of the *dual sampling method* that decides how many samples are required to view an area of the boundary surface and was developed in contrast to the *dual sampling method*. By solving the SCP after sampling had been performed, it resulted in a reduction of the number of ray traces and

collisions checks required to determine a valid viewing position in $\mathbb{R}^3$ and its resultant coverage in comparison to the *watchman route algorithm* of Danner and Kavraki (2000) (Englot and Hover, 2017). However, between the two stages of planning, Englot and Hover (2017) noted that solving the MPP was the more expensive process of the decoupled approach.

A different two-phase approach was taken by Bircher et al. (2015). The authors proposed the *iterative viewpoint resampling inspection planner* which maintains the separation of sampling and path planning but performs them iteratively, rather than in two distinct steps. The motivation behind this approach stems from using a sampling-based coverage approach to generate an inspection plan that acquires the coverage over the edges of the plan rather than the vertices. The authors suggest that generating a plan for a continuous sensing application, the number of viewing locations is not important, but how they are arranged in the configurations space is important. This decision removes any need to solve the SCP. Their iterative approach samples a set of viewing locations and using a TSP, the process continues to optimise the position of the viewing locations to minimise the length of the plan until the running time of the algorithm expires.

Jing et al. (2017a; 2017b) proposed another type of two-phase approach which separates the SCP problem from the sampling-phase and solves it concurrently with the MPP. Their approach used a *Random-Key Genetic Algorithm* to solve the *Set-Covering-(Generalised)-TSP* to better optimise costs associated with performing an inspection with high-DOF robotic platforms. By solving SCP concurrently with the MPP allows for flexibility to optimise the solution than a traditional decoupled approach that solved the SCP prior to solving the MPP (Chen and Li, 2004).

Coupled Coverage Planning Approaches

A shortcoming of the decoupled sampling-based coverage planning approaches is that they are predominantly implemented for robotic platforms that possess holonomic capabilities. Papadopoulos, Kurniawati, and Patrikalakis (2013) highlights that robotic platforms which do not possess holonomic capability, may find it difficult to achieve all the positions generated by the sampling process. Furthermore, as the sampling and path planning processes are performed independently, the costs associated with generating viewing locations and travelling between locations are not considered together for an optimised planning process (Wang et al., 2007); an important consideration for robotic platforms that

perform inspection under energy and time constraints. As such, *coupled approaches* that combine the sampling and path planning processes exist to overcome the limitations of *decoupled sampling-based approaches* to be applicable for a wider variety of inspection tasks and robotic platforms.

Papadopoulos et al. (2013) proposed the *Random Inspection Tree Algorithm* (RITA), a sampling-based inspection planner that simultaneously computes the viewing locations and plans the feasible trajectories for an AUV with non-holonomic constraints. Kafka, Faigl, and Váňa (2016) extended RITA to factor visibility constraints performed on a multi-legged robot. As the sampling and path planning procedures are executed concurrently, planning times for these methods are expensive.

An approach to reduce the computational expense of coupling sampling and path generation is to exploit the fast sampling features of sampling-based path planners like RRTs An example such as Bircher et al. (2017), proposed the *rapidly exploring random tree of trees* (RRTOT) to create fast inspection plans using the principles of sampling-based path planning. Utilising the properties of RRT*, a feasible inspection plan is found first by growing sub-trees over the entirety of the environment, then iteratively refining a path to create an admissible inspection plan. A comparison to the work of Papadopoulos et al. (2013), despite differences in mobility constraints, found that RRTOT provided significant computational advantages for coupled planning approaches and was applicable to the mobility constraints of both holonomic and non-holonomic platforms in both 2D and 3D environments.

2.3.3 Online Coverage Path Planning Approaches

Unlike path planners that seek to incrementally find a path through an unknown environment or replan an existing path to accommodate unforeseen disruptions, online coverage path planners must achieve the same objective but, additionally ensure complete coverage is maintained over all new and existing surfaces. Achieving this objective is dependent on how much information about the environment is known prior to inspection. If the environment is completely unknown, a path through must be developed in real-time. Otherwise, as seen with path replanning strategies, if the environment is assumed to be known, with the potential that the environment will be different, the replanning strategies will be required to adapt the current plan to overcome the unforeseen circumstances.

The following sections review online coverage planning methodologies that acquire full

coverage to digitally reconstruct unknown and partially known environments. As discussed in Section 1.4.2, online planning methodologies heavily rely upon a mapping system to construct a representation of the environment that planning algorithms can use to plan upon. As mapping is not within the scope of this thesis, this section will not address how mapping is performed but how path planning algorithms are used to aid the construction of a map.

Exploration and Coverage Planning in Unknown Environments

Generating coverage plans over unknown environments requires the solution to the *exploration problem*. Thrun, Burgard and Fox (2006) defined the *exploration problem* as the problem of how to move a robot through an unknown environment so it can maximise its own understanding of the environment. The result of solving the exploration problem, is either a partially or complete representation of the previously unknown environment, making it suitable to use for underwater exploration (Hernández et al., 2019).

Techniques such as *Simultaneous Localisation And Mapping* (SLAM) are commonly used, to simultaneously localise the robotic platform within a map of the environment which the algorithm is concurrently constructing. However, SLAM only provides the mechanism by which a representation of the environment can be constructed, therefore solving one part of the *exploration problem*, but does not provide the robot with a mechanism to safely explore the environment (González-Baños and Latombe, 2002). To plan the robot's movements safely through an unknown environment requires the use of *exploration algorithms*, such as *Next-Best-View* (NBV; Connolly, 1985).

NBV algorithms seek to move the robot to unobserved locations within the environment by analysing the known free space and then directing the robot to move towards the *frontier*, the region of unexplored space that maximises the new coverage along the path (Banta et al., 2000; Juliá, Gil, and Reinoso, 2012). These approaches continually construct a plan in real-time as the robot explores the environment. Each time the robot moves to a new location the process repeats until no more frontiers remain or the objective of the mission has been met. For path planning applications, *exploration algorithms* terminate when the goal is met regardless of how much of the environment was observed. However, for coverage planning applications, the role of *exploration algorithms* is to generate a feasible path through the environment that aids in the discovery of as many surfaces of the unknown environment as possible.

Fundamentally, the *exploration problem* and *inspection problem* can be used to solve the same problem, in that both attempt to capture the entirety of the environment. The difference between these two problems is the quality of inspection that is required. If all that is required is a complete representation of an unknown 2D or 3D environment, solving the *exploration problem* is sufficient. However, if the surfaces also need to be observed by a visual sensor for high-fidelity inspection, a *coverage planner* is typically required to work in tandem with an *exploration planner* to provide sufficient high-quality coverage of the surfaces (Song and Jo, 2018).

Generating a high-quality visual inspection of an unknown environment can be achieved by solving each problem in a two-phase approach. An example of a two-phase approach was presented in Bircher et al. (2018). The authors approach generated a representation of the unknown environment using the *receding horizon next best view exploration planner* (Bircher et al., 2016) before applying the offline *iterative viewpoint resampling inspection planner* (Bircher et al., 2015) to create a high-quality surface rendering of the previously unknown environment. A ROS implementation of this online coverage planning approach was presented in Papachristos et al. (2019).

Alternatively, the high-quality visual inspection of an unknown environment can be achieved in a single-phase process. These approaches construct a plan that combines the paths required for exploration with the paths required to satisfy high-quality coverage of discovered surfaces (Acer and Choset, 2002; Heng et al., 2015; Song and Jo, 2018). The approach by Acer and Choset (2002) used *Morse decompositions* to determine critical points in an evolving environment in real-time to create new cellular decompositions that can be covered uniformly using the *cycling algorithm*. Song and Jo (2018) employed the sampling-based coverage planning approaches formulated by Englot and Hover (2017) to create an inspection plan that incorporated viewing locations that simultaneously sought to acquire maximum coverage of the unknown surfaces and a high-quality inspection of the same surfaces.

Exploration after Coverage Planning

When the initial environment is assumed to be known before execution, solving the *exploration problem* first is not necessary. Coverage plans can be generated offline and executed as demonstrated previously. However, the idealised plans generated by an offline planner may not account for positional and sensor-based uncertainty, or generate a plan

based on an approximated representation about the environment. These examples could result in an inspection that is not sufficiently able to cover the entire environment, resulting in gaps in the generated representation of the environment. In these circumstances, *exploration algorithms* can be used post inspection to find new paths to fill these gaps instead of recomputing a new coverage plan to cover these regions. Utilising *exploration algorithms* after an offline plan has been executed minimises the planning efforts that online planning algorithms are required to perform. Online paths are only required to be generated to fill the gaps of the partially known environment rather than for the entire environment, as an online only solution would require.

Blaer and Allen (2009) implemented a two-phase coverage planning approach for a mobile platform acquiring coverage to create 3D digital rendering of significantly large buildings. The first phase generated an offline coverage plan, using an extended variant of the *dual sampling algorithm*. This algorithm was applied over a 2D floor plan of a 3D structure. After the initial inspection was conducted, the collected scans created a voxelised representation of the 3D environment. Processing the voxelised representation, highlighted the gaps that were then filled in the second phase of planning using an NBV algorithm.

Hernández et al., (2017) applied the two-phase approach to fill the residual gaps generated from a predefined survey of underwater structures. The difference between their approach to the one presented by Blaer and Allen (2009) was that the identification and generation of new paths to fill the gaps was an automated process that occurred immediately after the initial survey was completed. By combining the two processes into a single automated mission provided the robot the ability to complete a full inspection without human intervention, therefore reducing the time it takes to complete inspection tasks of a partially known environment.

Concurrent Replanning of an Existing Coverage Plan

An alternative approach to online coverage planning is to rectify the coverage plan immediately as new information about the environment is presented to the robot. Similar to the previous examples, when the environment is assumed to be known in advance, a robot is given a predefined plan to complete the inspection. If unforeseen obstacles are encountered throughout the inspection that may hinder the progression of the robot, the current plan must be adapted accordingly. However, instead of replanning to overcome the obstacle, a coverage plan must also provide sufficient coverage of the obstacle, so that it is

present in the final map reconstruction. Unlike exploration approaches that create an inspection plan on the fly, these methods use replanning strategies to adapt an existing plan to include coverage of newly detected changes within the environment when they become present to the robot. Therefore, at the conclusion of the inspection, all surfaces of the environment have been covered and the robot would not be required to perform an additional inspection phase.

Galceran et al. (2015) proposes a continuous *adaptive coverage planner* to overcome new features and address position uncertainty when executing a pre-calculated nominal path. STOMP (*stochastic trajectory optimization for motion planning algorithm*; Kalakrishnan et al., 2011) is used as a *plan repair* technique to resolve impeded trajectories due to newly detected protruding underwater structures. The iterative coverage planner reshapes a path segment of the original path yet to be executed by the AUV to provide a smooth trajectory around the feature that maintains the optimal viewing distance for coverage. The term *plan repair* is used in this context because segments of the plan are incrementally considered for replanning, even though the entire plan could have been replanned at the conclusion of the inspection.

Nykolaychuk and Ortmeier (2015) also uses a *plan repair-like strategy* to locally adapt an existing plan to reprocess defective areas for surface treatment and quality inspection applications. Upon detection of a surface defect, potential fields (Khatib, 1986) are used to augment the current plan around the defect. Attractive potentials draw positions of the initial path closer to the defect to obtain the coverage that is required. The results found that when the defect is detected close to the inspecting tools, the attractive potentials have minimal deformation on the inspection plan which ensures full coverage can be maintained. However, for defects detected further away, significant deformation of the plan occurred that could not ensure full coverage.

For multi-robot inspections, replanning coverage for a changing environment can be achieved by reallocating the new coverage amongst all the robots used for the inspection task. Given the presence of new or removed features or the removal of a robot from the inspection task, Williams and Burdick (2006) use a *plan revision strategy* to redistribute the remaining unobserved coverage amongst the remaining robots in the planning problem. The adaptation of an existing graph structure allows unobserved coverage to be reallocated or ignored.

2.3.4 Coverage Path Planning Summary

In summary, this section has provided an overview of CPP algorithms that create single continuous collision-free plans to inspect and traverse 2D and 3D environments. As with path planning approaches, common CPP approaches can also be divided into cellular and sampling-based approaches. For cellular based coverage, the objective is to achieve uniform coverage of a surface by either traversing over the surfaces or at an offset distance. Sampling-based coverage planners that built upon the foundations of *view planning* are suitable for the application of inspection where the robot can acquire high-quality coverage along a path comprising discrete locations. Each of these approaches have their benefits and limitations. The choice of which approach is most appropriate is determined by the platform and the sensor suite available to perform the coverage task. However, Galceran and Carreras (2013) conclude in their survey that, as *sampling-based coverage approaches* provide coverage plans for a wide variety of environments, these approaches are considered state-of-the-art in coverage planning across complex 3D structures.

To address the *online requirements* of the STIPP criteria, approaches to online coverage planning in both completely unknown and partially known environments were also explored. A review of online techniques that employ *exploration algorithms* running on autonomous platforms to perform coverage planning in unknown environments revealed the following approaches:

1) generate coverage plans simultaneously when exploring the environment,
2) utilise exploration algorithms after the initial offline coverage plan has been completed, and
3) replan the current plan to include newly detected surfaces.

Solving the *exploration problem*, the discovery of new surfaces in a partially explored environment, is not covered within the scope of this thesis; the reader is referred to the survey by Chen, Li and Kwok (2011) for more examples of established *exploration algorithms* and techniques planning within completely unknown environments. The following section discusses how the techniques reviewed in the literature survey relate to the STIPP criteria, and thus develop the gap in the literature that is addressed in this thesis.

2.4 Consolidation of the Literature to Resolve the STIPP

The purpose of this literature review was to determine which coverage path planning

methodologies were applicable to solve the STIPP. These methodologies are derived from the principles of path planning and consequently the review into the path planning literature which highlighted the importance of the environmental representation. Two common planning methodologies were discussed to solve a path by either;

1) decomposing the environment into graph representation, or
2) sampling a path in the configuration space.

The application of these techniques is the basis of similar techniques found in the CPP literature. A review into CPP algorithms demonstrated how the representation of the environment can be used to generate a variety different coverage plans. From the review, *cellular decompositions* tend to dominate the solution space for coverage plans that rely upon the *traversable coverage* or uniform coverage of a surface. *Sampling-based coverage planning* relies upon the combination of view and path planning techniques, and tend to generate sets of discrete view locations to create coverage plans that observe the boundary surfaces of 2D and 3D geometries. A simple distinction between these two methodologies can be determined by how this coverage is acquired. *Continuous coverage planners* tend to acquire coverage over the edges of a coverage plan, while *discrete coverage planners* acquire coverage on the vertices of a plan.

To investigate the *online criteria* of the STIPP, replanning methodologies were explored, in both the path planning and CPP literature. When an existing plan becomes compromised, path replanning strategies are used to resolve the compromised segments of the plan. Typical path replanning strategies will perform either a *full replan*, *partial replan,* or a *plan repair* to rectify compromised paths. The important distinction between these strategies is defined by how much of the existing plan is preserved during the replanning process.

When the environment is unknown prior, *exploration algorithms* are required to continually create and revise paths in real-time through the unexplored regions of a changing environment until the goal in achieved. For CPP, *exploration algorithms* can be used in tandem with an *inspection planner*, to explore and provide high-quality coverage of unknown environments. When conducting inspections of partially known environments, gaps that form in partially constructed representations can be filled in a two-step process where an *exploration algorithm*, configured to actively find paths to areas that have not been captured, can be executed after the initial offline plan has been undertaken. Alternatively, to avoid the two-step process, newly detected changes in the environment can be replanned

into the current inspection plan. These approaches use path replanning strategies to ensure the new coverage is included in the final coverage plan.

Given the STIPP criteria and the reviewed literature, it was decided that discrete sampling-based CPP methodologies were best suited to satisfying the *foundational requirements* of the STIPP, since sampling-based approaches;

1) use discrete viewing locations to acquire coverage,
2) do not directly parameterise the environment to generate coverage,
3) attempt to attain the highest possible coverage, and
4) can generate coverage plans for robotic platforms that possess high DOF.

While there is strong coupling between path and motion planning for *continuous* CPP techniques, constructing a *continuous coverage plan* may prove to be too computationally expensive to generate a plan for multi-legged platforms in complex environments. This is particularly the case if these plans need to be revised and adapted in real-time. Furthermore, while it is possible to segment a sweeping trajectory with regular intervals, whether it be based on distance or time (Galceran et al., 2015), the main issue behind using continuous CPP techniques is *how these approaches decompose the environment to produce the coverage trajectories*.

The applicability of some of these decomposition approaches in a complex environment may occlude or exclude narrow and confined spaces. Environmental decompositions, especially in the case of 2.5D representations, reduce the dimensionality of the space to make it easier to create feasible coverage plans. The approaches that decompose the space could potentially compromise the accuracy of the representation of the environment, putting at risk the ability of the covering planning algorithm to cover all areas of the environment (Englot and Hover, 2012). Given that the role of the *inspection planning module* is to generate a coverage plan that attempts to completely cover all the surfaces, regardless of the optimality of the path, the best effort should be given to provide high-quality inspection without the risk of inadvertently missing coverage due to approximating the representation of the environment.

Coverage, or lack thereof, acquired by sampling-based approaches is determined by how well the robot constraints enable the robot to observe the environment. If the coverage planner is unable to find an adequate viewing location to observe a particular region, the loss of coverage will not be due to the coverage planner's approximation of the environment, but

an inability of the robot platform to move to a suitable viewing position due to the complexities of the environment or the constraints of the robot. As sampling-based approaches can satisfy these *foundation requirements*, it is suitable to employ sampling-based path and coverage path planning techniques to construct an existing inspection plan to satisfy the *offline requirement* of the STIPP (*Requirement 1;* Section 1.5.1).

The only contention to using a sampling-based approach is that it may not meet *Requirement 5*. Consideration must be given to the complexities and issues that do arise from using a multi-legged platform. Section 2.3.2 discussed how robotic platforms of complex mobility constraints, namely non-holonomic platforms, may suffer from the typical two stage approach to sampling-based coverage planning. Techniques were developed to couple the sampling and paths processes, but they were also proven to be computationally expensive.

Despite the concept demonstrator possessing holonomic capabilities, given the mobility constraints of the multi-legged platform chosen, motion planning will be computationally expensive. A benefit of the sampling-based coverage planning approach that uses a decoupled approach is that any type of motion or path planner can be used for the second phase of planning. Therefore, regarding the IPF, it may be possible to integrate the *motion planning module* within the *inspection planning module* to resolve path planning queries. Furthermore, sampling-based path planning methodologies, such as RRTs, are best used to resolve path planning queries for a platform of such complexity. The use of RRTs for adaptive path planning is intended to be used on the concept demonstrator (Pivetta et al., 2017). However, further discussion about the integration of the multi-legged platform into the IPF is presented in *Chapter 3*.

As the environment of the submarine tanks are known *a priori*, a complete online solution that utilises *exploration algorithms* is not required. An offline *sampling-based coverage planner* will derive an initial inspection plan from the geometrical representation of the known environment. *Exploration algorithms* would be a necessary component within the IPF, however, implementing such an approach is the responsibility of the *mapping system module* (Section 1.5.3). The *inspection planning module* is responsible for replanning the newly detected coverage. It is also not desirable for the inspection to take place over two planning phases. Given the complexities associated with planning multi-legged platforms, performing the two phases to complete the inspection will result in an extended time to complete the inspection.

To satisfy the *online requirements*, a desirable approach to account for new changes would be to replan new coverage into the existing inspection plan as it is received, therefore minimising the time required to perform a complete inspection. However, for online adaptive coverage planning methodologies that adapt an existing plan to provide coverage for new features, there are few algorithms that solve either continuous or discrete CPP problems. The algorithms that were reviewed were based upon techniques that have already been discussed and found to be unsuitable to meet the STIPP criteria.

2.5 Gap Statement

To the author's knowledge at the time of writing this review there were no adaptive sampling-based coverage planners developed that concurrently replan new coverage into an existing plan. Since there was no current solution available that meets all the requirements STIPP, especially the online requirements, a gap in the literature was identified.

This thesis addresses the gap by developing an *adaptive sampling-based coverage planner* that incorporates a replanning strategy with the offline planning processes to solve the STIPP. Since prior knowledge of the environment is available before planning, it is appropriate to adapt a discrete offline coverage planner that is capable of solving the initial inspection plan over the target environment. To handle the case when the initial plan is violated due to changes in the environment, the offline coverage planner can be coupled with a path replanning strategy. Such a strategy converts the *offline coverage planner* into an online *adaptive coverage planner*.

The thesis herein presents the methodology and findings that formulated the creation of a new adaptive coverage planner that addresses the gap in the literature highlighted by the STIPP criteria. Emphasis of this thesis is focussed on understanding the intrinsic behaviours of a chosen offline coverage planner in a submarine tank-like environment, so the online variant that utilises the underlying offline processes can be modified to perform effectively when coupled with a replanning strategy. The analysis provides evidence to support the choice in replanning strategy chosen to extend the procedures of an offline coverage planner to work as an efficient adaptive coverage planner that can provide coverage in partially known complex environments. To the author's knowledge the new adaptive coverage planner and subsequent algorithms presented in this thesis to address the gap in the literature have not been attempted before or have been implemented for the use within a submarine ballast tank.

2.6 Chapter Summary

In this chapter a literature review was presented that focussed on offline and online path and CPP approaches, with the goal of discovering if existing approaches had the capability to solve the STIPP criteria. The review found that there were no immediate coverage planning solutions that completely satisfied the STIPP criteria. Of the techniques that were reviewed, it was determined that discrete *sampling-based coverage planning* algorithms were best suited to solve the STIPP. However, as there were no online sampling-based coverage planners that could concurrently replan an existing plan, whilst performing an inspection through a partially known environment, a *gap statement* was formulated.

To address the gap statement, and consequently meet all the *foundational*, *offline* and *online requirements* of the STIPP, it was determined that an *adaptive coverage planner* should be developed from an *offline sampling-based coverage planner* that is capable of generating coverage plans within submarine tanks for a multi-legged platform. An approach that is adopted for this thesis is to extend the capability of an *offline coverage planner* with a *path replanning strategy* to create the desired *adaptive coverage planner*. In the following chapter, a solution is formulated to solve the STIPP using this approach.

Chapter 3

Formulation of a Solution for the STIPP

3.1 Introduction

Having defined the *gap statement* in *Chapter 2*, this chapter presents a formulation of a solution for solving the *submarine tank inspection planning problem* (STIPP; Section 1.5.1). The *gap statement* identified that there are no current online implementations of *discrete coverage planners* that modify an existing plan to account for newly identified features that would meet all the criteria specified to solve the STIPP. It was therefore concluded that since *discrete sample-based coverage planners* are met the *foundational requirements* of the STIPP criteria, to address the *online requirements* of the STIPP, it would be best to adapt an *offline coverage planner* to work in the online domain.

This chapter explains the selection of an appropriate *offline sampling-based coverage planner* that meets the *offline requirements* of the STIPP criteria. The *offline coverage planner* is presented in further detail, with its known limitations identified as potential concerns for an online implementation. Of these concerns, two were highlighted that will be addressed in this thesis. Next, two path replanning strategies are proposed that are suitable to adapt the *offline coverage planner* to work in the online domain. This chapter concludes with a discussion about the assumptions placed on the relationships between the *mapping system*, *inspection planner* and *motion planner modules* of the *Inspection Planning Framework* (IPF; Section 1.4.4). These assumptions constrain the focus of the thesis to exclusively investigate the intrinsic properties of the *offline sampling-based coverage planner* as it is adapted for online implementation.

3.2 Selection of an Appropriate Coverage Planner to Solve the STIPP

Given the STIPP criteria, the *sampling-based coverage planner* developed by Englot and Hover (2011, 2017) that uses *redundant roadmaps* to provide coverage over a complex environment, proved to be the best candidate to solve the offline planning problem and provides a suitable base to create an online adaptive variant to solve the STIPP and address the gap statement provided in *Chapter 2*.

Most importantly, the *offline sampling-based coverage planner* is a *discrete planner* that generates a series of individual static viewing locations that enables the robot to stabilise at a given position to take photographs of the 3D environment (*Requirements 1 and 2*). As discussed in *Chapter 2*, a *discrete coverage planner* is better suited to solve the STIPP as image consistency is of particular importance.

Like most discrete coverage planning algorithms, the *offline sampling-based coverage planner* generates coverage based on the constraints of the sensor and not on the explicit parameterisation of the environment. Therefore, the coverage planner is capable of being applied generically across multiple tank layouts and variations (*Requirement 3*). The results in Englot (2012) highlight the ability of the coverage planner to provide inspection plans over various ship hull layouts.

Given the generic nature of the coverage planner, it will always seek to achieve the highest attainable coverage over a complex environment (*Requirement 4*). The *redundant roadmap*, which is used to acquire the coverage of the boundary surface, continues to sample view positions until all primitives of a mesh have been viewed for a *redundant number of times*. To ensure the completeness of the random sampling methodology, Englot and Hover (2012a, 2013) presented the first probabilistic completeness proof for a *sampling-based coverage planner*. Complete coverage can be assured providing the *prison cells* are avoided. *Prison cells* are important to consider, especially when planning online, and therefore are discussed in further detail later in the thesis (Sections 4.2.2 and 7.2.4).

The separation of the coverage generation and path planning algorithms make it easier to accommodate a multi-legged, high-DOF robot (*Requirement 5*). While the details of the *sampling-based coverage planner* are discussed in Section 3.3, the motion planning of a high-DOF robot can be treated as a separate process, enabling the separation of high-fidelity motion planning from the *inspection planning module*. This thesis distinguishes between

path and motion planning for high-DOF robots, by simplifying the constraints used to represent the robotic platform. Simplified visibility and mobility constraints are applied to the *inspection planning module* to provide the *motion planning module* with the paths that can then be solved to a higher fidelity. Details surrounding the platform constraints in this thesis are discussed in Section 3.6.2.

The final two requirements of the STIPP focus on the online aspect of creating an *online adaptive coverage planner*. Given the coverage planner is indeed an offline implementation, it inherently does not have the immediate capabilities to operate online. However, in Englot (2012) it was proposed, as future work, that the planning processes behind the *sampling-based coverage planner* could be applied to the online domain, when combined with *active perception* (Bajcsy, 1998). Englot suggested that these planning processes can operate as *anytime algorithms* to modify an existing coverage plan if unexpected events were to occur. Knowing that these planning processes could operate as *anytime algorithms*, makes them suitable for online implementation, especially when the timing of planning updates is critical. *Despite this recommendation, no further suggestions on how to implement this extension were provided.*

While the planning processes have been identified as suitable for *anytime implementation*, the modularity of the *sampling-based planner* provides a framework that lends itself to be easily implemented for online adaptation, therefore addressing *Requirement 6*. As each process acts independently to one another, each process can be modified and improved without impacting the outcomes of the other. A sampling-based approach that couples the two processes may not enable individual modifications of each process to be easily achievable. Furthermore, as the computational costs are expected to be expensive for a multi-legged robot, handling these planning processes independently will aid in reducing the costs of replanning a robot of such computational expense. Sections 3.4.3 and 3.6 provide more details on how this was achieved.

The output of the planner also makes it simpler to adapt coverage plans online. The output of the coverage planner comprises of a collection of piecewise path segments. These piecewise segments can be easily segmented based on either the geometry or influence of the changes. The ability to segment a plan at any point enables any of the three replanning strategies discussed in Section 2.2.3 to easily modify the current plan (Figure 3-1).

The remaining challenge is to decide which replanning strategy is most appropriate to satisfy

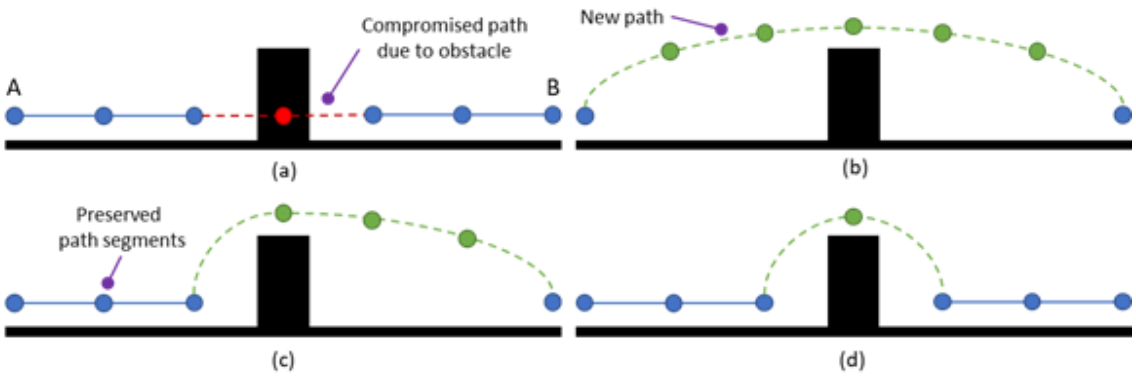


Figure 3-1: Different approaches to replan a piecewise inspection plan to changing conditions. (a) The current plan is compromised due to the detection of a new obstacle. (b) *Full replan strategy*. (c) *Partial replan strategy*. (d) *Plan repair strategy*.

the *gap statement* and hence meet *Requirement 7* of the STIPP. While there are online implementations that apply the sampling and planning procedures of this *sampling-based coverage planner* in an unknown environment (Song and Jo, [2017, 2018]), to the author's knowledge, *there are no extensions that explicitly apply a replanning strategy over the offline procedures to replan an already existing plan in partially known environments*. Therefore, choosing the most appropriate replanning strategy is subject to further investigation. The remainder of this thesis focusses on applying the most suitable replanning strategy that will enable an online implementation of the *offline sampling-based coverage planner* to deliver planning updates in a timely manner to solve STIPP.

3.3 Offline Sampling-Based Coverage Path Planning Using Redundant Roadmaps

The *offline sampling-based coverage planner* solves the coverage problem by solving two sub-problems in series. It first solves the *coverage sampling problem* (CSP) by constructing a *redundant roadmap* to observe the boundary structure. This is followed by solving the *multi-goal planning problem* (MPP) to produce a *feasible collision-free inspection plan* that ensures, within geometric limitations, 100% of the boundary structure is covered. Figure 3-2 summarises the planning procedures of the *offline sampling-based coverage planner* developed by Englot and Hover (2017).

In the following sections, the primary components of the *offline sampling-based coverage planner*, the CSP, which includes a description of how the *set cover problem* (SCP) is used and solved, and the MPP are discussed in detail. Figure 3-3 provides illustrative examples of the *offline sampling-based coverage planner* procedures over a representative I-beam

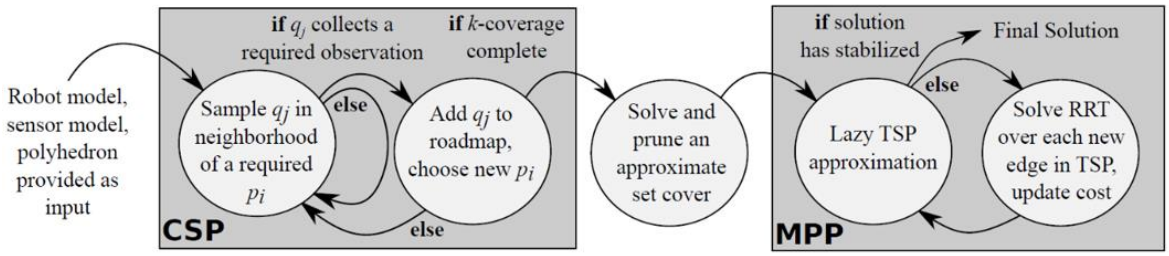


Figure 3-2: The state-flow diagram demonstrating the offline sampling-based coverage planning algorithm presented (Englot, 2012; reproduced with permission).

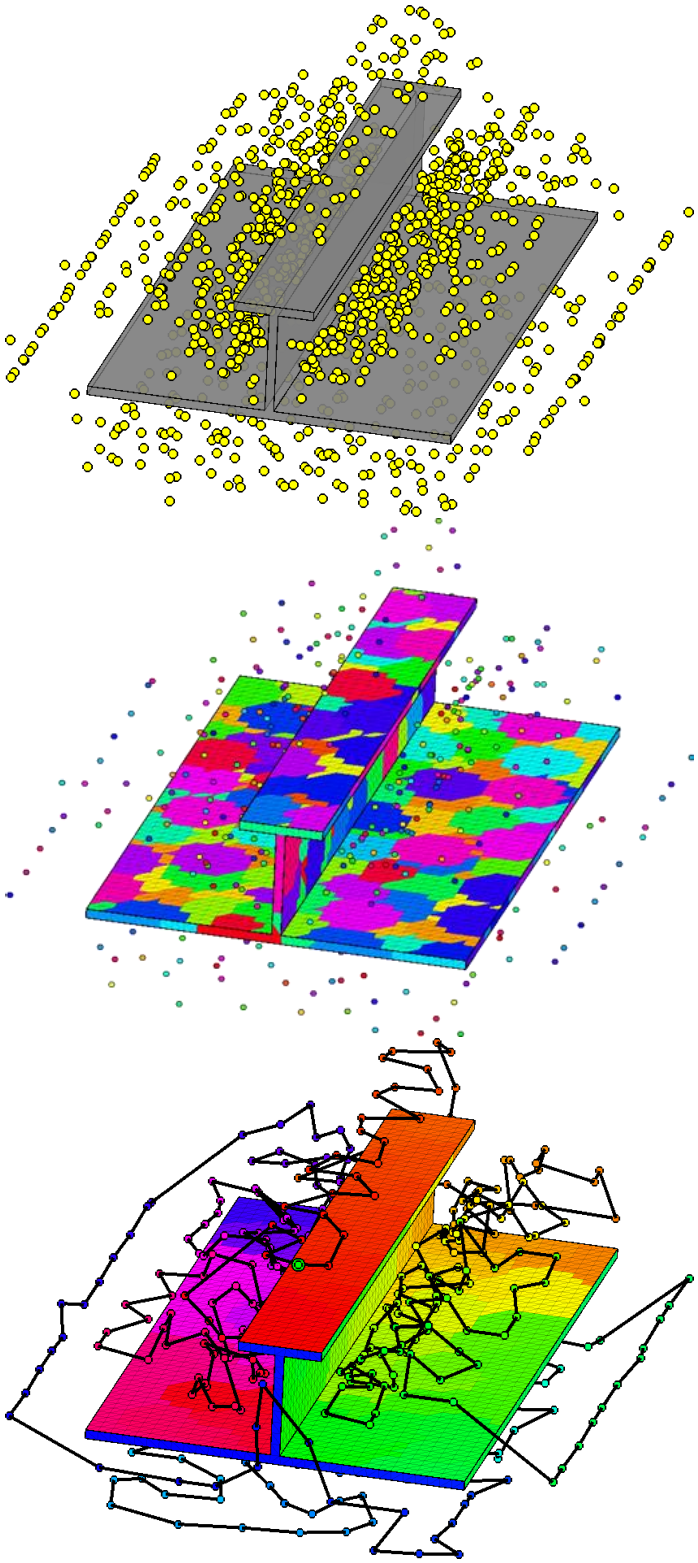
structure that is commonly found in the inside a submarine tank. Figure 3-3 will be referred to throughout these sections to provide a visual explanation of each of these procedures.

From this point onwards in the thesis, the term *configuration* is used interchangably with the terms *viewing location* and *viewing position* to represent the 6-DOF position the robot must obtain in $\mathbb{R}^3$ to achieve the desired calculated coverage of a surface. The details on how the *offline coverage planner* solves coverage plans for a 18-DOF hexapod robot with 6-DOF in $\mathbb{R}^3$ is discussed in Section 3.6. Furthermore, the terms *inspection plan*, *plan* and *tour* are all used interchangably to describe the collection of collision-free paths that connect all the *configurations* in the inspecton plan as a result of solving the MPP.

3.3.1 Solving the Coverage Sampling Problem

As illustrated in Figure 3-2, the CSP is responsible for building the *redundant roadmap* to cover the boundary structure. When analysing the probabilistic completeness of the CSP, Englot and Hover (2012a) defines the CSP as a *set system* (P, Q) (Haussler and Welzl, 1987; Isler, Kannan, and Daniilidis, 2004). In short, this *set system* (P, Q) enables a subset of primitives in the *primitive space* (P) to map to a single configuration in the *configuration space* (Q) i.e. what primitives, $p_i \subset P$, are observable from the configuration, $q_j \in Q$. An illustrative representation of the *set system* (P, Q) is presented in Figure 3-4. Using a *set system* representation, Englot and Hover (2012a) defined the CSP as follows:

> **Definition 1** (Coverage Sampling Problem): *Let P be a finite set of discrete geometric primitives p_i comprising a structure to be inspected. Let the infinite set of Q be the robot configuration space whose configurations q_j of Q map to observations of the Euclidean workspace which contains P. Let integer k be the number of times each $p_i \in P$ must be viewed. Find a set of feasible configurations $N \subseteq Q$, that obtains at least k distinct views of all $p_i \in P$.*



(a) A *redundant roadmap* is constructed to cover all primitives of the mesh a *k*-redundant number of times to solve the *coverage sampling problem*.

(b) The *set cover problem* reduces the *redundant roadmap* to a minimal set of configurations that cover each primitive at least once. Each configuration in this image is colour coded to its respective coverage on the surface.

(c) The *lazy point-to-point planner* is used to solve *multi-goal planning problem* over the reduced *redundant roadmap*. Colour in this image shows the direction of travel starting from the green configuration.

Figure 3-3: An offline coverage plan solved over a representative 1-metre I-beam structure using the offline sampling-based coverage planner.

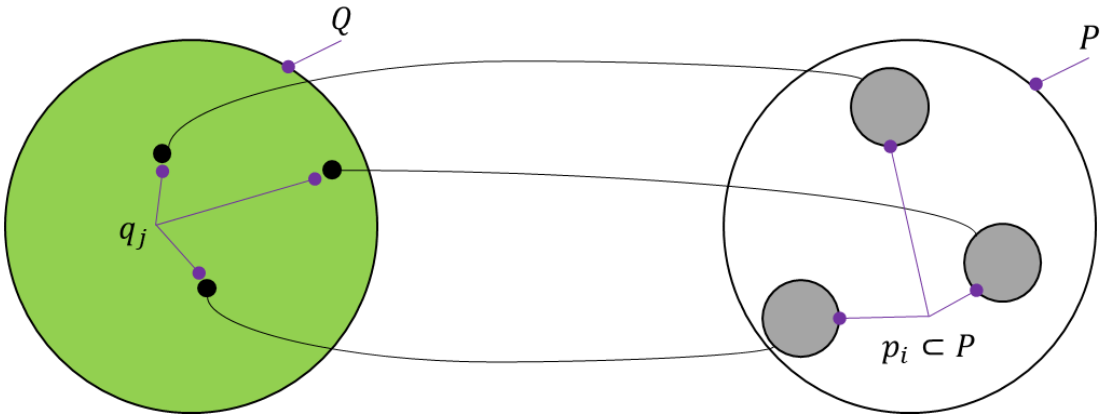


Figure 3-4: The set system (P, Q) and the relationship between the continuous configuration space Q and discrete primitive space P. The set system highlights the mapping of a configuration q_j to a subset of primitives $p_i \subset P$.

In all of Englot and Hover's publications the boundary structure was constructed from a triangular mesh and each triangle of the mesh was considered a primitive. This thesis also uses triangular meshes to represent boundary structures of the environment. The use of the term primitive ensures the application of this definition can be applied to any structure that is represented by a finite set of elements.

Algorithm 3-1 presents a more detailed account of the CSP process. For completeness, Algorithm 3-1 is replicated from the last known publication on the *offline sampling-based coverage planner*, Englot and Hover (2013). Construction of a *redundant roadmap* begins with the initialisation of all the primitives that compromise the boundary structure to be listed as unobserved (Line 1). An *unobserved primitive* that has yet to have its redundancy fulfilled is selected at random (Line 4). To observe this primitive, a robot configuration is then sampled at random within the local neighbour of primitive (Line 5), akin to the *dual sampling method* (Gonzalez-Banos and Latombe, 1998; González-Baños, 2001) to avoid sampling exhaustively in areas of the workspace that may not guarantee coverage of the selected primitive.

Figure 3-5 illustrates how configurations are sampled within a local neighbourhood based on the visibility constraints of the sensor. Sampling a configuration within the local neighbourhood, of the selected primitive, ensures that the calculated coverage for this configuration will observe the primitive with the exception of occlusion by other geometries.

Once a configuration is sampled, a series of collision and visibility checks are performed to verify that the sampled configuration is suitable to obtain the calculated coverage (Line 6).

Algorithm 3-1: Coverage Sampling Problem (After Englot and Hover, 2013).

```
CoveringSet = BuildRedudantRoadmap(Primitives, Obstacles, Redundancy)
 1:  UnobservedPrimitives ← Primitives
 2:  CoveringSet ← Ø
 3:  while UnobservedPrimitives ≠ Ø do
 4:    Primitive ← ChooseRandamPrimitive(UnobservedPrimitives);
 5:    NewConfig ← SampleFeasibleConfiguration(Primitive,  Obstacles);
 6:    Coverage ← CalculateCoverage(NewConfig, Primitives,  Obstacles);
 7:    NeededSightings ← Coverage ∩ UnobservedPrimitives;
 8:    if NeedSightings ≠ Ø then
 9:      CoveringSet.add(NewConfig, NewSightings);
10:      for i ∈ NeedSightings do
11:        NeededSightings[i].incrementRedudancy();
12:        if NeededSightings[i].RedundancyCount() = Redundancy then
13:          UnobservedPrimitives ← UnobservedPrimitives \ NeededSightings[i];
14:        end if
15:      end for
16:    end if
17:  end while
18:  return CoveringSet
```

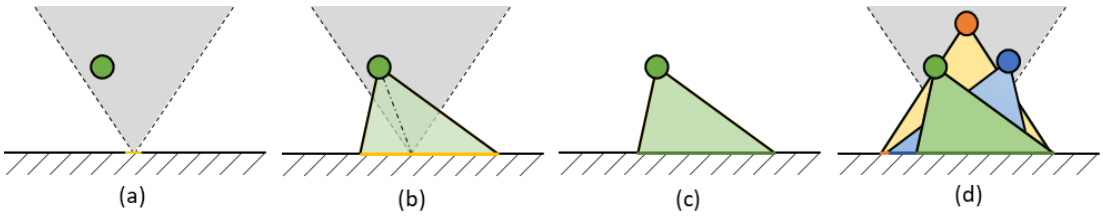


Figure 3-5: Generating configurations. (a) Configurations are drawn within the local neighbourhood that are defined by the envelope of the sensor parameters. (b) The coverage is calculated, when the sensor parameters are projected onto the surface. (c) The primitive along with neighbouring primitives within the field of view of the sensor become observed. (d) Within this envelope the redundancy can be met by other samples.

If the configuration is deemed valid for the robot to achieve and the configuration is able to observe at least one primitive on the surface that is required to be seen (Line 7), the configuration is added to the *covering set N* (Line 8) and the redundancy of those primitives are incremented (Lines 10-11). If any of these primitives have met their redundancy due to the new coverage, they are removed from the list of *unobserved primitives* (Lines 12-13). This process continues until the redundancy of all primitives of the boundary surface have been met. Figure 3-3a shows the application of the CSP over the representative I-beam structure.

The defining feature of the *redundant roadmap* is that the redundancy of the roadmap is a tuneable parameter that is set by the user. The quality of the final solution can be tailored to the desired outcome. Englot and Hover (2017) show that roadmaps created from higher

redundancies results in overall inspection tours that contains less configurations and are shorter than roadmaps constructed with a lower redundancy, at the expense of longer sampling times. In their experiments, diminishing returns on tour quality, both for the number of configurations and tour lengths, were observed in *redundant roadmaps* with redundancies greater than ten. Therefore, ten samples per primitive was considered a *redundant roadmap* of high-quality.

3.3.2 Solving the Set Cover Problem

The SCP is solved using a *polynomial-time greedy approximation algorithm* (Chvatal, 1976) to reduce the size of the *redundant roadmap* to a minimum cardinality set (S) that covers each primitive of the environment at least once. The greedy algorithm, on each iteration adds to S, the configuration with the largest number of sighted primitives. Solutions obtained using the greedy approach are bounded to $ln(m) + 1$ of optimality, where m is the number of primitives in the problem. Given a greedy approximation is a mere approximation of the optimal minimal covering set, further redundancy can be removed from S.

To further remove redundancy, another pruning procedure was applied over S. The *iterative pruning procedure* randomly removes any configurations that do not uniquely observe any geometric primitives. The removal process continues until all configurations that remain in N view at least one unique geometric primitive. The pruning procedure used by Englot and Hover, (2013, 2017) ran in $O(n^2m)$ time, where n is the number of observations in the planning problem. Figure 3-3b shows the resultant *minimal covering set* that has been reduced by solving the SCP over the *redundant roadmap* in Figure 3-3a.

3.3.3 Solving the Multi-goal Planning Problem

The final stage of the *offline sampling-based coverage planner* is to solve the MPP to produce a single continuous collision-free path that connects the configurations in S. Englot and Hover (2012a) defined the MPP procedure as follows:

> **Definition 2** (Multi-goal Planning Problem): *Let $G \subset Q$ be a finite set of robot configurations which comprise the set of goals selected for traversal. Find a set of feasible paths in Q that joins all goals into a single connected component.*

To avoid solving all possible path permutations and then finding the optimal solution, a *lazy point-to-point planner* (LPP), inspired by Saha and Latombe, (2003) was used to solve the MPP. The adapted variant presented in Englot and Hover (2013) is shown in Algorithm 3-2.

Algorithm 3-2: Multi-goal Planning Problem (After Englot and Hover 2013).

```
RobotTour = LazyPoint2PointPlanner(S, Obstacles)
 1:  AdjMat ← EuclideanDistances(S);
 2:  UnclearedEdges ← GetEdgePairs(S);
 3:  ClearedEdges ← Ø;
 4:  while NewTourCost ≠ PreviousTourCost do
 5:    PreviousTourCost ← NewTourCost;
 6:    NewTourCost ← 0;
 7:    LazyTour ← ComputeTour_TSP(AdjMat);
 8:    for Edge_ij ∈ LazyTour do
 9:      if Edge_ij ∈ UnclearedEdges then
10:        FeasiblePath_ij ← SolvePath_RRT(Edge_ij, Obstacles);
11:        ClearedEdges ← ClearedEdges ∪ Edge_ij;
12:        UnclearedEdges ← UnclearedEdges \ Edge_ij;
13:        AdjMat(i,j) ← PathCost(FeasiblePath_ij);
14:      end if
15:      NewTourCost ← NewTourCost + AdjMat(i,j);
16:    end for
17:  end while
18:  RobotTour ← LazyTour
19:  return RobotTour
```

The LPP solves the complex path planning problem by iteratively solving a *Travelling Salesman Problem* (TSP) to find the shortest tour amongst all configurations to then have those edges explicitly solved using a motion planner. The LPP achieves this by firstly initialising the adjacency between all configurations to be *Euclidean* (Line 1). Upon each iteration, the TSP solver approximates the shortest tour (Line 7). For each edge selected by the TSP solver, if it has not already been resolved in a previous iteration, it is explicitly solved using a path planner (Lines 8 - 12).

The distances received from the edges solved by the path planner are updated in the adjacency matrix (Line 13) and collectively form the overall cost of the tour for that iteration (Line 15). The LPP continues to iterate through the solution space, updating the adjacency between configurations until the TSP solver produces identical solutions in successive iterations (Line 4). When this condition is met, the LPP terminates and provides the edges of the tour, found in the last iteration, as the final solution. Figure 3-3c shows the result of the LPP over the minimal covering set solved by the SCP in Figure 3-3b.

3.4 Algorithmic Concerns

While the *offline sampling-based coverage planner* was identified as the state-of-the-art in Galceran and Cararras (2013), there were some concerns that have been identified in the

literature that need to be considered or addressed. Concerns arising in the literature are as follows;

1) factoring the uncertainty of the mesh and sensing model,
2) applicability for decoupled planning processes invalidating robots of restricted planning constraints,
3) complete coverage may not be obtainable in the real world, and
4) long computational times associated with the LPP.

These concerns may degrade the performance of the algorithm online. This section discusses the listed issues and how they are addressed in the thesis. The conclusion of this section highlights the pertinent issues that could potentially hinder the advancement of the *offline sampling-based coverage planner* from being converted to work online.

3.4.1 Uncertainty of the Mesh, Sensing and Motion Models

Galceran et at. (2015) highlighted that the *offline sampling-based coverage planner* does not handle uncertainty in the mesh and sensor models when planning. Given that this thesis presents simulated results, factoring uncertainty into the early stages of algorithmic development was not a priority. Therefore, for the purposes of constraining the scope of the thesis to just the development of an *adaptive coverage planner*, uncertainty of the mesh, sensor and motion error is not investigated in this thesis. This work assumes zero sensing or motion errors across all mapping, coverage and motion planning modules of the IPF. However, while the uncertainty is a serious concern, especially for real-world implementation, once an *adaptive coverage planner* is formulated, the inclusion of uncertainty into the planning problem should naturally be the next extension to improve upon the quality of the solutions.

3.4.2 Applicability of a Decoupled Solver for Non-holonomic Robotic Platforms

As discussed in the literature review (Section 2.3.2), a *sampling-based coverage planner* that separates the coverage and path planning stages runs the risk that the final solution may not be applicable for all types of non-holonomic platforms (Papadopoulos et al., 2013; Kafka et al., 2016; Bircher et al., 2017). As the *adaptive sampling-based coverage planner* is not designed to couple the sampling and path planning processes, it will be acknowledged that it may not be applicable to all robotic platforms.

While the future capability of the concept demonstrator will be restricted to move only along

the surfaces, the robotic platform does possess holonomic capabilities. The individual leg control enables the robot to move in all directions. Given that the motion planning of the 18-DOF platform is not the focus of this thesis, this thesis will assume that the robotic platform will possess the mobility capabilities and constraints of an *unmanned aerial vehicle* (UAV) or *autonomous underwater vehicles* (AUV) that can hover at a desired position. This assumption allows it to be possible to simplify the motion planning constraints of the robot while ensuring the robot can reach the calculated positions. The details of how this assumption is applied within the coverage planner and still ensures coverage plans are applicable for an 18-DOF robot is explained in Section 3.6.

3.4.3 Complex Geometry that Invalidates Complete Coverage

While complete coverage is desirable, Ellefsen, Lepikson and Albiez (2017) highlight that complete coverage, especially in complex environments, is not always achievable. Despite the *probabilistic completeness* of the coverage planner, platform mobility constraints and complex geometries could possibly mean that full coverage may not be practically achievable. This problem is likely to compound when new features are detected online. Partially detected features have the potential to exclude existing structures and impede viable positions that acquire coverage.

This concern was also discussed when presenting the STIPP criteria (Section 1.5.1). *Requirement 4* requires that the coverage planner achieve the highest attainable coverage given the robotic platform constraints and complex geometry. Given that in reality complete coverage may not be achievable, it was acceptable to relax the 100% complete coverage constraint, provided an attempt had been made to cover the challenging regions, and if they cannot be covered appropriately ensure they are reported to the human operator either during or after the inspection. The method by which *unobservable primitives* are captured and tracked is explored in this thesis (Section 4.2.3; Section 7.5.2).

3.4.4 Long Computational Time Associated with LPP

The final concern raised is that the *offline sampling-based coverage planner* is computationally expensive. Nykolaychuk and Ortmeier (2015) highlight that this *offline coverage planner* would not be feasible to implement in a replanning context due to the time taken to construct a new plan. To run this coverage planner effectively online, the long solution times must be resolved. For an offline implementation a high execution time for a pre-calculated solution is acceptable. However, for a responsive online solution, the response

of a coverage planner to produce solutions in a reasonable time is critical.

Englot and Hover (2017) identified that the LPP contributes significantly to the overall planning times. However, no indication was provided regarding the cause. One may assume it is due to the $O(n^2)$ nature of the LPP. However, Saha and Latombe (2003) claim that their implementation never got close to this bound in practice.

For large planning problems it is appropriate to assume that the larger the covering set, the more edges need to be resolved and if the LPP is having difficulty finding a solution in a complex space, the motion planner could be called several hundreds of times per planning iteration. Therefore, the time it takes to find a solution could accumulate quite significantly over a long time. One way to resolve this problem would be minimise the number of calls made to the motion planner. However, regardless of the potential reasons that causes the LPP to exhibit large computational times, this problem needs to be addressed, if not mitigated, to ensure the LPP is capable of performing effectively online.

3.5 Online Adaptation of the Offline Sampling-based Coverage Planner to Resolve the Online Requirements of the STIPP Criteria

To resolve the *gap statement* and the *online requirements* of the STIPP criteria, a replanning strategy is required to adapt the offline coverage planner to work online. Three replanning strategies were discussed in Section 2.2.3 that are used to resolve compromised paths in changing conditions. These strategies were:

1) *a full replan*
2) *a partial replan*, or
3) *a plan repair.*

Of the three replanning strategies, this thesis will implement only two, the *full replan strategy* and the *plan repair strategy*. The reason for not deciding to implement a *partial replan strategy* is that the only difference between a *partial replan* and a *full replan* is where the planning updates occur. Each replanning strategy needs to handle the incremental nature of objects forming in the environment as they are detected over time (Figure 3-6). It is expected that mapping updates will not be presented as complete features and can occur anywhere throughout the environment. As there may be several map updates, not necessarily being received in succession, if replanning were to occur immediately the difference between the *partial replan* and *full replan* would be negligible, especially over plans that contain

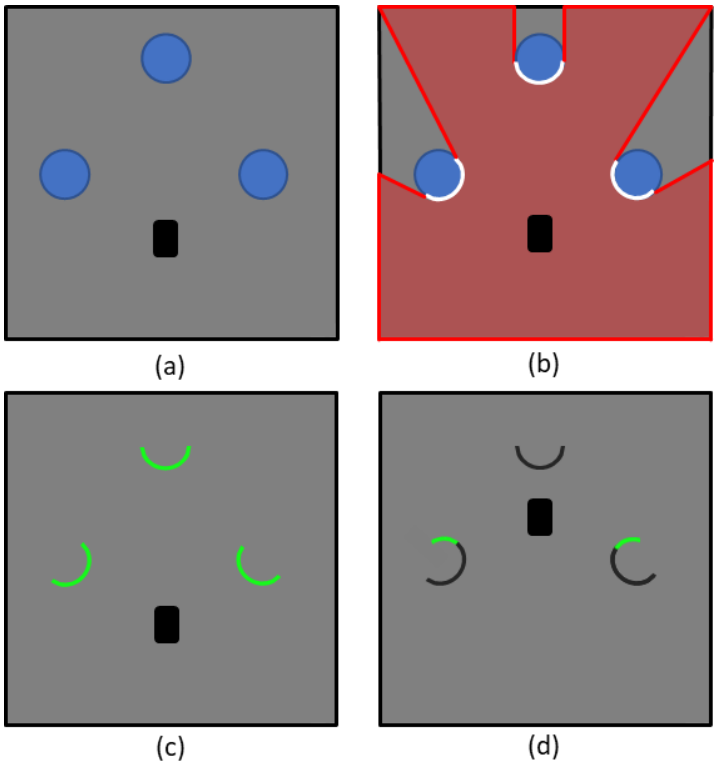


Figure 3-6: Identifying new features within a partially known environment. (a) A robot (black) has three unknown features within the *a priori* environment. (b) After a full LIDAR scan (red), the mapping system detects the partial outline of the three features that do not correlate to the expected surfaces (white). (c) The *mapping system* registers the changes and appends the partial information known about these features into the map (green). (d) When the robot moves forward new information is gathered about these features. The partial features will become whole once the robot explores more of the environment.

many configurations. A *plan repair strategy*, on the other hand, preserves as much of the uninfluenced segments of the plan regardless of the robot's current location.

The construction of the *adaptive sampling-based coverage planner* aims to preserve the decoupled planning processes given that their decoupling is not expected to impact the planning of a multi-legged platform (Section 3.2). Furthermore, maintaining the modularity of these planning processes makes it easier to create a replanning framework and to isolate individual processes for improvements without impacting other procedures.

While the sampling and path planning modules are easily modifiable to perform as *anytime algorithms*, the constructed frameworks will not rely upon the *anytime* implementation to perform an online plan update. The construction of the online framework in this work will produce full and complete plan updates. Therefore, the full extent of each adaptive coverage planner can be analysed to determine the better approach. Once this can be established, these algorithms can be implemented as *anytime algorithms* to speed up the process of the planning updates. However, this is the subject of future work.

The *offline sampling-based coverage planner* has not been previously adapted to work as an online planner to modify an existing plan in partially known environments. This thesis, herein, investigates which of the two replanning strategies is best suited to extend the *offline sampling-based coverage planner* to perform online. The following sections proposes two *adaptive sampling-based coverage planners* that implement a *full replan* and *plan repair strategy* before discussing how the limitations of the *offline coverage planner* discussed in Section 3.4 will be addressed in this thesis.

3.5.1 Strategy 1: Full Replan

A tractable and reliable approach to replanning is to perform a *full replan* upon detecting features in the environment. A *full replan* will remove all the viewing locations and associated paths that are yet to be achieved by the robot to provide a new plan that incorporates the coverage of both the new and unobserved regions throughout the environment.

The benefit of performing a *full replan*, is that this strategy considers the plan globally, instead of locally like a *plan repair strategy*. This creates final plans that have been reassessed against the entire geometry of the environment and therefore should be shorter in length compared to those generated by a *path repair strategy*. Given the constraints of the target platform, minimising tour length will be beneficial. Furthermore, as the remaining tour is removed, there are less overheads associated with the *full replan strategy* as no exhaustive cross checks are required to determine which viewing locations and path segments remain preserved and which are required to be replanned.

Adopting a *full replan strategy* also ensures the minimal amount of change to adapt the *offline sampling-based planner* to perform adaptively online. As the remaining coverage is reassessed, the problem that is supplied to the *adaptive coverage planner* is essentially an offline problem. The sampling and path planning processes remain the same, just the new plan starts from the robot's current location and only plans for the uninspected area of the environment.

When the planning objective is simple and replanning is not frequent, the *full replan* is an effective strategy. However, with uncertainty surrounding the frequency and impact of any new features, regular replanning updates of partial surfaces will become expensive especially when replanning larger spaces, or small and compact complex environments. If

the replanning effort becomes too expensive it may outweigh the benefit of producing shorter plans than a *plan repair strategy* or the ability to simply create an *online adaptive planner* with minimal modifications to the existing architecture.

Consideration also needs to be given to the type of robotic platform that the adaptive coverage planner is replanning for when using a *full replan strategy*. Generating motion plans for the multi-legged platform is expected to be expensive. If existing paths and configurations are going to be continually removed and replaced, this will only increase the computation time associated with calculating new motion plans online.

Section 3.6.2 discusses how motion planning is handled in this thesis. However, to minimise the computational effort of computing high-fidelity motion planning within the *adaptive coverage planner*, the high-fidelity constraints of the multi-legged platform are replaced with simplified constraints. In short, high-fidelity motion planning will be exclusively performed by the *motion planning module* in the IPF after the coverage plan has been calculated. The plan produced by the *adaptive coverage planner* using these simplified constraints will provide a proxy plan of traversal for the high-fidelity motion planner to solve. Therefore, if the *adaptive coverage planner*, under a *full replan strategy*, continually replans a completely new plan, the expensive aspect of the overall planning effort will be shifted to the *motion planning module*.

To be an efficient *adaptive coverage planner*, it needs to reduce motion planning across the IPF. It would be ineffective to reduce the planning times of the *adaptive coverage planner* only to increase the computation of another module. However, if the overall replanning effort for the coverage and motion planner could be minimised so the computational load proves to be acceptable for online use, implementation of a *full replan strategy* may be a suitable approach to extend the *offline sampling-based coverage planner* to work adaptively online.

3.5.2 Strategy 2: Plan Repair

The second strategy minimises the replanning effort of the CSP and MPP by performing a *plan repair* over segments of the plan that have been impacted by change. Compared to the *full replan* strategy, the *plan repair strategy* is a riskier approach to implement. There are more processes to be considered, as the *plan repair strategy segments*, *replans* and *merges* new and current coverage plans together.

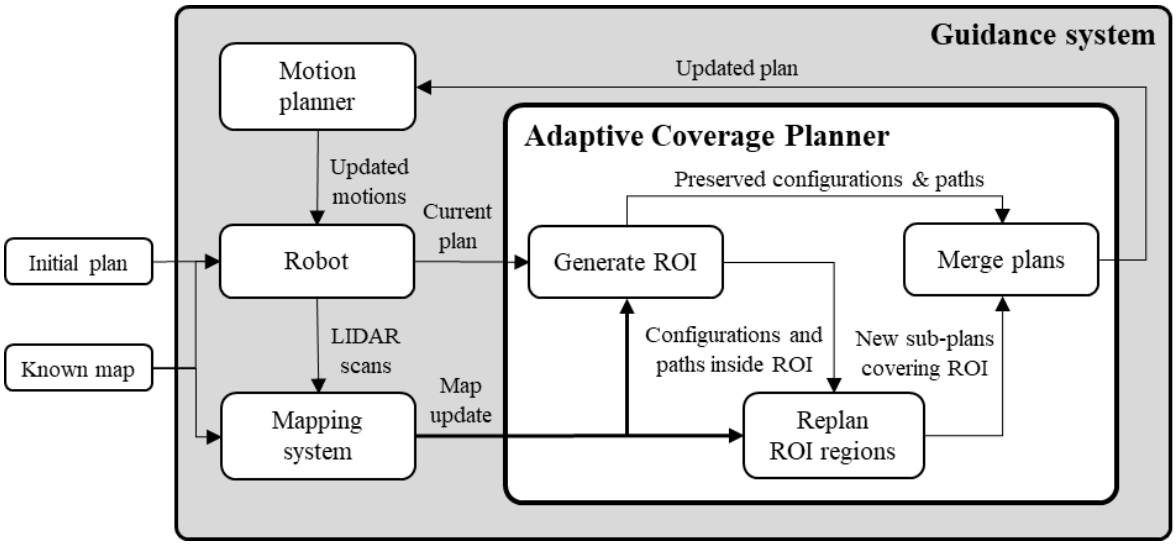


Figure 3-7: The proposed *adaptive sampling-based coverage planner* using a *path repair strategy* inside the *Inspection Planning Framework* (IPF). The *adaptive coverage planner* generates a *region of interest* (ROI) to segment the current plan into *preserved* and *unpreserved segments*. *Unpreserved segments* are replanned to cover the new features and are merged back into the current plan. The updated plan is handed to the motion planner to update the respective motion plans for the segments of the tour that were replanned. This process continues for each map update until the inspection plan is complete.

Figure 3-7 illustrates how the *adaptive sampling-based coverage planner*, using a *path repair strategy*, would be incorporated into the IPF. Upon receiving a map update from the mapping system, the current plan is segmented by *regions of interest* (ROIs) that encapsulate the influence the new features would have on the existing tour (Figure 3-8).

All the configurations and paths that exist within these ROIs become candidates for replanning while those that are outside the ROIs, are preserved. As the ROIs encapsulate the influence that the new features have within the environment, the replanning effort is only exclusive to the ROIs. The *plan repair strategy* proceeds to replan the coverage within the ROIs using the *offline* sampling procedures ensuring that all primitives, new and original, are covered before solving the MPP with each ROI.

Each replanned ROI creates a sub-plan that has been solved to observe the new features and provide a feasible path for the robot to traverse around. The final step merges the new sub-plans with the preserved portions of the existing plan to form an updated plan. This updated plan is then passed to the motion planner where the respective motions of the robot are updated. The *adaptive coverage planner* continues this process for each new change supplied by the mapping system until the inspection plan is complete.

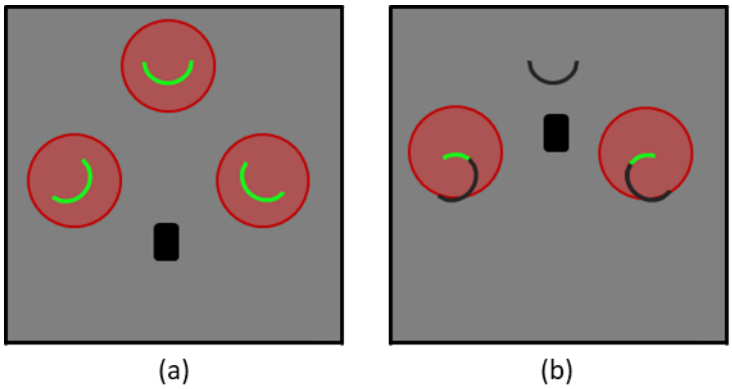


Figure 3-8: Bounding new features in a *region of interest* (ROI). (a) When the mapping system detects change, a ROI can be produced around each detected feature. (b) When the robot moves through the environment and gains more information about the environment, new ROIs encompass the new changes.

The benefit of implementing a *plan repair strategy* over a *full replan strategy* is that segmenting a plan based on influence will always ensure the size of the problem will scale to the size of the amount of change, or the size of the influence, of those changes. Only if the changes influence enough of the existing tour, the *plan repair strategy* will be equivalent, if not worse, than replanning the entire environment again.

While the *path repair strategy* may present as the faster solution, it can only be considered potentially faster during the replanning phase, when the sampling and path planning procedures are being used. Smaller coverage problems should replan faster as fewer configurations will need to be sampled thus resulting in fewer motion paths that need to be evaluated. However, a *plan repair strategy* has more to consider when formulating a solution. Unlike a *full replan strategy*, a *partial replan strategy* has to;

1) determine where the changes occur within the environment,
2) isolate a suitable ROI that appropriately encapsulates the influence the changes have on the existing tour,
3) segment the current tour so uninfluenced regions are not replanned,
4) have the ability to replan several unconnected regions within environment, and
5) ensure the plan is merged correctly and remains a single continuous tour upon completion of the update.

These requirements introduce additional costs to the replanning costs associated with the two planning processes of the *offline coverage planner*. These additional costs, to *segment*, *replan* and *merge*, could potentially degrade the computational efficiency that a *plan repair strategy* may provide. Considering *a plan repair strategy* replans locally, and not globally

like the *full replan strategy*, it has the potential to have a negative impact on tour quality. Similar to how the *full replan strategy* will pass the costs of high-fidelity motion planning down to the motion planner in the IPF, if the *plan repair strategy* significantly increases the tour length, the cost of executing the tour may outweigh the benefits of solving it faster. However, in the case of a complex submarine tank environment, where there could be up to several hundreds to thousands of randomly placed configurations, preservation in computation would be a reasonable trade-off for a suitable degradation in tour quality.

3.5.3 Considerations of Replanning Strategies

For the robot to perform effectively online, it requires an *adaptive coverage planner* that can provide stable, consistent and timely updates. Given that both *adaptive coverage planners* are directly reliant upon the performance of the existing CSP and MPP algorithms, it is important to understand how the *offline sampling-based coverage planner* performs in the target environment.

As discussed in Section 3.4, there are two primary concerns that were raised about the offline sampling and path planning procedures that need to be addressed to ensure they do not impact online performance. These concerns were;

Concern 1) to ensure that all known primitives are accounted for at the conclusion of the inspection whether they be observed or not (Section 3.4.3), and

Concern 2) the LPP is known to consume the most planning time of all processes of the coverage planner (Section 3.4.4).

Of the two concerns, *Concern 2* is likely to severely influence online planning times, given the reported impact the LPP has already had on offline planning times. Addressing the impact of *Concern 1* can be achieved quite simply by appropriately tracking which primitives have or have not met redundancy. While the *adaptive coverage planner* that uses a *plan repair strategy* is expected to perform less replanning than a coverage planner using *full replan strategy*, the impact of the LPP will be felt, regardless of which approach is taken.

Given there are prior results available on the performance of the *offline sampling-based coverage planner* inside a submarine tank, the following chapter performs a benchmark experiment to acquire these results. The objectives of the benchmark experiment are:

Objective 1) Benchmark the performance of the *offline sampling-based coverage planner* within a representative submarine tank environment.

Objective 2) Determine to what extent the *lazy point-to-point planner* (LPP) influences the planning times within the target environment.

Objective 3) Discover any other limitations of the *sampling-based coverage planner* that may impact online performance.

The outcomes of the benchmark experiment aid in determining;

Outcome 1) whether the offline sampling and planning procedures are capable of performing effectively online,

Outcome 2) provide an indication as to which of the two replanning strategies is better suited for online implementation, and

Outcome 3) provide insight into any potential improvements that could increase the performance of the sampling and path planning processes for online implementation.

In the following section, the extrinsic constraints that would influence both the *adaptive* and *offline sampling-based coverage planner* are minimised. T*he focus of this thesis is the analysis and development of a generic adaptive sampling-based coverage planner that can operate regardless of the type of robotic platform, mapping system or motion planner.* Constraints are placed upon the interactions between the coverage planner and the other IPF modules to limit the impact that these modules have on the intrinsic properties of the generic coverage planner so the analysis of the planning modules can be compartmentalised. Solving this problem under these constraints is applicable to solve the STIPP.

3.6 Limiting Computational Effort by Reducing Extrinsic Planning Constraints

There are a number of extrinsic constraints that are placed on both the *offline sampling-based coverage planner* and the online adaptive variant which can significantly influence the computation times. These include;

1) how the robot detects features within the environment (Mapping),
2) the ability of the robot to move within the environment (Motion planning),
3) the ability of the camera to observe the surfaces of the environment (Visibility constraints) and,
4) the properties and complexity of the environment.

Of the listed extrinsic constraints, the constraints placed upon a robot's manoeuvrability and observability have a greater impact upon the algorithm's performance. A true reflection of

the coverage planner capabilities is to minimise the number of manoeuvrability and visibility constraints placed on the robot, so the environment is the main extrinsic factor that influences the planning problem. Understanding the coverage planner's response to varying levels of environmental complexity allows the intrinsic behaviour of the coverage planner to be the focus of the analysis.

In this thesis, a *holistic approach* was adopted to evaluate the performance of both the *offline sampling-based coverage planner* and the *online adaptive variant*. The approach taken isolates the coverage planner from the physical realisation of the robot platform or any specific mapping system. *Removing the explicit properties of the robot from the analysis ensures that any improvements made to algorithmic properties of the coverage planner will be applicable to robotic platform constraints of any complexity.* More importantly, the insights will provide a better understanding of which improvements will aid the algorithms to perform effectively online, irrespective to the chosen robotic platform.

The following sections elaborate on the assumptions made in Section 1.5.2 as to how the modules of the IPF are connected in this thesis. The relationship between the environment, mapping system, and *adaptive coverage planner* is discussed. Furthermore, an explanation is provided for the reduction of the two primary constraints *visibility* and *mobility*, which are associated with any given robotic platform performing an inspection task. These assumptions serve to limit the impact that motion planning and the visibility constraints would exert on the performance of the coverage planner.

3.6.1 Assumptions Placed on the Relationships between the Environment, Mapping System and Adaptive Coverage Planner

As the mapping system and *adaptive sampling-based coverage planner* were developed independently, to ensure these two modules integrate together for implementation of the concept demonstrator, a few assumptions were placed on the relationship between the environment, mapping system and *adaptive coverage planner*. The list of assumptions used to bind the relationship between the three components were as follows:

Assumption 1: Newly detected features all exist within the bounds of the original environment.

New features are expected to occlude and prohibit movement through regions of the original environment that may have been originally possible to traverse and observe. It is expected that the *adaptive coverage planner* will account for primitives occluded by the new features.

How the *adaptive coverage planner* handles occluded primitives is discussed in Sections 4.2.3 and 7.5.2.

Assumption 2: Changes within the environment are expected to be small in relation to the size and complexity of the environment.

It is reasonable to assume that the environment may deviate from nominal operating conditions by no more than 30%. Certain structures within the submarine tanks are either added or removed as a part of the inspection and cleaning process. With respect to the size of the environment, these changes are expected to be small in nature and may be scattered throughout the environment.

Assumption 3: The mapping system will only introduce new features into the environment. Given the nature of online mapping, partial maps are also expected.

For the purposes of practicality, the inspection planning problem only handles the introduction of new features. Moved or removed features require consistent tracking of primitive indices by both the mapping and coverage planner to ensure that coverage is always maintained.

As the *adaptive coverage planner* was developed independently to the mapping system, the assumption that only new features would be introduced was sufficient enough to allow these modules to interact during real-world experiments without being explicitly coupled together. Therefore, the mapping system and coverage planner both maintained complete but individual versions of the environment. To ensure primitive indices remained the same throughout the inspection, both the mapping system and the coverage planner were initialised with the same offline map that was used to create the offline plan. As only new changes were added to the map, any new primitives were just appended to the map. Therefore, upon detection, the new complete map was then passed from the mapping system to coverage planner with a threshold index (m_{index}) to indicate where the new primitives began. This assumption was the basis of how the *adaptive coverage planner* worked in practice. Details to this are explained in *Chapter 7*.

While implementing a centralised mapping system that all IPF modules can access is ideal, it is not implemented in this work and is a subject for future work. Furthermore, whilst modified and removed primitives are not considered in this thesis, the principles behind both replanning strategies presented in this thesis will still ensure these primitives that modified

or moved primitives would still remain covered at the end of each replanning iteration.

Assumption 4: No features within the environment are dynamic in nature.

Any new features are static and will remain static throughout the lifetime of the inspection. The robot will be the only moving object in the inspection environment. This assumption simplifies the constraints placed on the planning problem. Once the mapping system confirms a new feature or original structure is present it will remain true for the lifetime of the inspection.

Assumption 5: The *adaptive sampling-based coverage planner* will not handle mesh, sensor and positional uncertainty.

Since the mapping system and *adaptive coverage planner* are separate modules in the IPF, it will be the responsibility of the mapping system to factor uncertainty of the mesh. To constrain the limits of the thesis, uncertainty of the camera model used for inspection is not included in the focus of this thesis. The coverage that is observed by a calculated pose is assumed to be reached and acquired.

Assumption 6: The *adaptive sampling-based coverage planner* is not responsible for uncovering or directing the robot to uncover new surfaces of partially constructed features.

The *adaptive coverage planner* will only provide an updated inspection plan for detected areas of the map that are unobserved. In reality, it is possible that the robot may not detect all surfaces of a new feature. The *adaptive coverage planner* is not responsible for finding a path to guide the robot to observe the undetected surfaces of the feature. This functionality is handled between the mapping system and guidance system. The mapping system may choose to indicate to the guidance system to temporarily halt the inspection plan to investigate, via next-best view techniques, to detect the remaining feature surfaces. How the mapping system determines the location of partial information and the algorithms involved to deviate the robot from the current inspection plan to acquire new information about the environment is not covered in this thesis and is subject for future work. Given an offline plan is used initially to guide the robot through the environment and that *Assumption 1* states that all information will be contained within the bounds of the original environment, this thesis will assume that all new features can be viewed along the original and updated inspection plan. Replanning can occur simultaneously to performing the inspection.

Given the sensing range of the LIDAR is significantly greater than the physical size of the robotic platform, the robot can detect new features well before the robot is capable of moving to those positions. Therefore, this enables the opportunity for replanning to occur in advance. While there is a possibility the robot may have to pause and replan immediate features that appear within close proximity to the platform, if replanning can be performed fast enough, the platform will not pause for each replanning update.

In summary, the *adaptive sampling-based coverage planner* is to only cover the new information presented to it by the mapping system. The complete coverage of new information and the tracking of occluded primitives is the sole focus of the adaptive variant. Applying these assumptions on the relationship between the environment, mapping system and coverage planner do not invalidate the STIPP criteria but constrain the scope to only focus on the adaptive aspect of the thesis that converts the *offline sampling-based coverage planner* to work effectivity online. These assumptions will be more prevalent when proposing and testing the *adaptive sampling-based planner* in *Chapters 7* and *8*.

3.6.2 The Separation of High-fidelity Motion Planning from the Coverage Planner

Motion planning is an inherit cost that is associated with any actuated autonomous robotic platform. Calculating an effective solution is dependent on the constraints placed on the planning problem. Generally, the more DOF within the planning problem the more expensive planning becomes to find a solution. While a multi-legged robot is a suitable robotic platform for tank inspection (Section 1.2), the mobility constraints placed on a multi-legged, high-DOF robotic platform will incur a significant computational cost when planning within a confined environment (Short and Bandyopadhyay, 2017).

A submarine tank is a highly structured and static environment. The submarine tanks and various internal structures prevent surface traversing using standard cycling gaits without sufficient ability to tolerate leg slip (Hörger, 2014). Motion planning within these environments will require sophisticated algorithms that manipulate each leg independently to reach calculated positions. Consequently, the mobility and environmental constraints will likely have a significant impact on the time it takes to generate a coverage plan.

As explained in Section 3.3.3, the *offline sampling-based coverage planner* explicitly couples together path and motion planning. When solving the MPP, the LPP acquires the true length of any given path between configurations by solving the motion plan based on the robotic platform's constraints. Given that the LPP has already been identified as the main

contributor to the coverage planner's long planning times (Section 3.4.4), applying the mobility constraints of a multi-legged platform, will only increase the LPP planning times. To significantly remove the impact of these mobility constraints, a simplification to the platform constraints will aid in reducing the impact motion planning will have on the coverage planner.

As this thesis is not concerned with improving the computational efficiencies of motion planning for high-DOF robot platforms, it is proposed to defer the high-fidelity motion planning of a multi-legged, high-DOF robot, which includes the positioning of the body and legs, until after a coverage plan has been developed. Applying simplified planning constraints to the coverage planner limits the influence motion planning has on generating a coverage plan. Solving the MPP with simplified constraints, enables the LPP to quickly find a feasible solution and delay the evaluation of high-fidelity motion planning until the coverage planner has developed a solution.

Motion planning still occurs inside the LPP. However, instead of solving the *high-fidelity paths*, that can be applied directly to a robot upon completion, these resultant paths, solved under simplified constraints, provide an ordered proxy tour that a high-fidelity motion planner can plan upon once complete. Delaying the evaluation of these paths reduces high-fidelity motion planning across the whole system, as it will only have to occur over the edges of the final coverage plan instead of over all the edges evaluated by the LPP.

Postponing the evaluation of the actual motion planning allows for both deliberate offline and reactive real-time motion planners to solve the point-to-point planning problem. This approach offers greater flexibility and may reduce the penalty of complex motion planning within the planning process. Separating the high-fidelity motion planning from the coverage planner fits into both the offline and online planning architectures of the IPF. These processes can be treated as separate entities and developed independently.

There are two notable consequences of separating high-fidelity motion planning from the coverage planner:

1) The tour generated by the coverage planner under simplified constraints is an estimation of the actual plan. The paths solved by the high-fidelity motion planner are likely to be longer in length.

2) Planning under simplified constraints opens up the possibility that the robot may not be physically able to achieve every viewing location. The ability to achieve all configurations is essential because each configuration provides unique coverage of the environment. A simplification of the visibility constraints, via the relaxation of pose constraints, will aid in making these positions achievable (Section 3.6.3).

These compromises are made to allow feasible solutions to be produced within a reasonable time. Until motion planning inside the LPP is computationally feasible, the separation of these two planning modules is acceptable despite the consequences of not comprehensively solving the inspection plan.

To ensure the robot has the best opportunity to achieve all the positions, the simplified constraints are constructed by isolating the positioning of the camera from the positioning of the entire multi-legged platform. The simplified planning constraints approximate the tibia of a 3-DOF robotic leg of an 18-DOF platform that contains the camera needed for visual inspection (Figure 3-9). A spherical collision model is used to represent the ankle joint and is given 6-DOF with holonomic mobility constraints to adequately position the camera in $\mathbb{R}^3$. The 6-DOF will represent a pose or robot configuration that consists of a 3D Cartesian position and three Euler angles: roll, pitch and yaw. Applying holonomic mobility constraints to solve the planning problem allows the results of this thesis to remain applicable to any robotic platform that is represented as a 6-DOF holonomic platform, such as UAV's and AUV's.

Applying a 6-DOF holonomic robot was taken over by the application of a simplified hexapod model as presented in Tonneau et al. (2018). Applying simplified constraints, represented the minimal free body collision model that can exist in $\mathbb{R}^3$ compared to other collision methods like Axis-Aligned Bounding Boxes (AABBs; LaValle, 2006), that will be used to represent the environments later in this thesis. Using a simplified and cheaply calculated collision model creates a lower bound to solving motion plans using the *sampling-based coverage planner*. It is expected that any representation of the robot greater than the spherical model will only increase planning times, but it does make the collision constraints easier to solve in an online context.

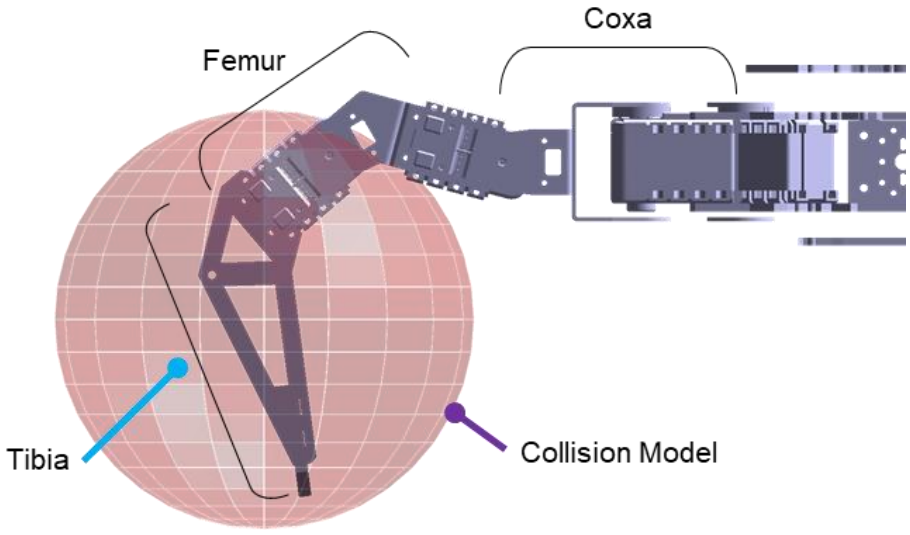


Figure 3-9: A spherical collision model is used to represent the ankle joint of a 3-DOF leg on a 18-DOF PhantomX robot containing the camera for inspection. The spherical model is given 6-DOF to find appropriate viewing locations. PhantomX MKIII model adapted from Trossen Robotics (2019). Reproduced with permission**.**

3.6.3 Simplifying Visibility Constraints to Enable High-Fidelity Motion Plans to Achieve Simplified Viewing Locations

Finding a valid viewing position requires all DOF and planning constraints to be satisfied within an environment. For a multi-legged robot, a valid position ensures the robot is statically stable. If any of the planning constraints are not met, the sampled position is rejected.

It is only after the 6-DOF coverage plan has been developed that the positions are evaluated to determine if they can be reached by the robot. Therefore, there is a possibility that some of the viewing locations may not be achievable when the full 18-DOF solution is generated. The coverage planner can generate achievable positions by relaxing the visibility constraints of the robot to enable the high-fidelity motion planner a better chance of achieving these positions.

Most images from a camera are rectangular in nature, with an aspect ratio of either 4:3 or 16:9. Achieving the calculated coverage of a camera explicitly requires constraining the roll, pitch and yaw angles. Any modification to the camera pose will result in a different outcome (Figure 3-10a). Given the complexity of the planning constraints and the environment, calculating a valid static position for a multi-legged platform that satisfies the geometric position and three explicit angles of a given camera pose may not be possible.

If a calculated position cannot be reached during execution, to achieve maximal coverage the high-fidelity motion planner may opt to modify the position to attain the best possible

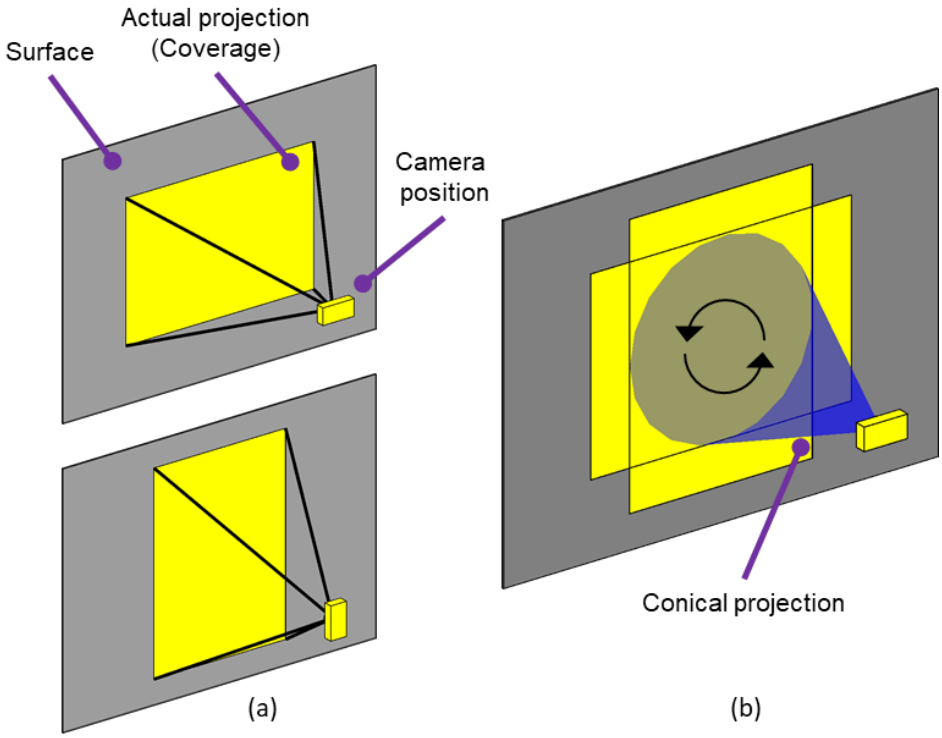


Figure 3-10: Representation of the actual and relaxed visibility constraints. (a) Coverage under rectangular projection constraints restrict the roll of the camera. (b) Assuming a circular projection that underestimates the rectangular projection allows for the calculated coverage to be viewed by any roll angle.

coverage. Modifying the pose under strict visibility constraints may result in areas of the surface that are left uncovered. To allow the high-fidelity motion planner the freedom to choose which rotational constraints provide the best opportunity for the robot to achieve the calculated poses, the visibility constraints supplied to the coverage planner are relaxed.

The relaxed visibility model assumes that the sensor casts a circular projection onto a surface (Figure 3-10b). Applying this circular projection, which underestimates the rectangular projection, allows the calculated coverage to be perceived as rotationally invariant whilst the actual observation taken by the rectangular projection observes more than calculated. This assumption removes the explicit roll angle constraint allowing the high-fidelity motion planner to decide which roll, pitch and yaw angles are appropriate, given all the planning constraints, to obtain the calculated coverage.

As the circular projection underestimates the actual coverage of the sensor, more viewing locations will be required to cover the surfaces of the environment. The increase in configuration density will create more overlap between acquired images than images obtained using the actual visibility constraints. Creating more overlap is not of concern, as the more overlap between images will assist in the image registration required to digitally

reconstruct the environment for external examination (Scott et al., 2003). The consequence of using these relaxed constraints is that there will be more configurations in the planning problem, therefore increasing the time to sample and plan. In order to minimise the time spent calculating configurations, and to accommodate the separation of the high-fidelity sensor model from the coverage planner, it is unavoidable that there will be an increase in the number of samples a planning problem contains.

3.6.4 Summary of Assumptions between the Coverage Planner and the Motion and Visibility Constraints

Continuing from the list of assumptions presented in Section 3.6.1, a summary of the discussion points presented in Sections 3.6.2 and 3.6.3 create assumptions that bound the expectations surrounding the relationship between the extrinsic constraints of a robotic platform and the coverage planner.

Assumption 8: A 6-DOF holonomic assumption is used as a heuristic to approximate the motion constraints of an 18-DOF robot with electromagnetic adhesion.

Assumption 9: High-fidelity motion planning will occur after the coverage plan has been calculated to reduce the impact on planning times. The *sampling-based coverage planner* will solve motion plans with the simplified constraints of *Assumption 8*. Therefore, the tour provided by the coverage planner approximates the actual tour.

Assumption 10: Visibility constraints are simplified to aid the high-fidelity motion planner to achieve the calculated 6-DOF by relaxing the roll constraint of the viewing pose.

Unless specified otherwise, Table 3-1 presents the simplified robotic platform constraints used to define the robot platform in the experiments undertaken in this thesis.

3.7 Chapter Summary

In the chapter, the *offline sampling-based coverage planner* of Englot and Hover (2017) which utilises *redundant roadmaps* to generate coverage over complex spaces was chosen as the subject for adaptation to solve the STIPP. This chapter discussed *the offline sampling-based coverage planner* in detail before highlighting documented limitations about the cover planner. Of the listed concerns discussed, two were identified that could

Table 3-1: Visibility and collision model constraints.

Parameter	Value
Field of View (FOV)	±45°
Minimum Field of Depth (FOD_{min})	100mm
Maximum Field of Depth (FOD_{max})	270mm
Collison sphere radius size:	50mm

potentially impact online performance. These concerns were:

Concern 1) to ensure that all known primitives are accounted for at the conclusion of the inspection whether they be observed or not, and

Concern 2) the LPP is known to consume the most planning time of all processes of the coverage planner.

To address the *gap statement* and to provide a solution to meet all the STIPP requirements, two replanning strategies were proposed to create an *adaptive sampling-based coverage planner*.

1) The first proposal was to use a *full replanning strategy*. *A full replanning strategy* would require minimal modification to the existing decoupled planning architecture as each map update can be treated as an *offline* planning problem.

2) The second proposal is to use *path repair strategy* to segment the space around changes to preserve and minimise the amount of computation required to perform an update. Only the new feature and influence regions will be replanned using the existing sampling techniques and the new paths that form within the regions will be merged into the existing plan.

While it is expected that the *full replan strategy* will likely be the more computationally expensive approach compared to the *path repair strategy*, given that the *path repair strategy* has additional processes to *segment* and *merge*, it may encounter computational overheads that may outweigh the benefits.

As the response of the *offline sampling-based planner* is unknown in the target environment, *Chapter 4* investigates the functionality of the *offline sampling-based coverage planner* over a series of planning environments, including a representative tank environment, to determine

if any other issues are present that may impact the online performance. From this chapter onwards, this thesis will use the assumption placed on the relationships between the environment, mapping system, motion planner and coverage planner so a thorough analysis of the intrinsic behaviours of the sampling and path planning procedures can be conducted within minimal influence of extrinsic constraints.

Chapter 4
Evaluating the Functionality of the Offline Sampling-Based Coverage Planner in Preparation for Online Planning

4.1 Introduction

In *Chapter 3*, two replanning strategies were proposed to extend the *offline sampling-based coverage planning* of Englot and Hover (2017). The first was a *full replan strategy* which would replan all the remaining uncovered regions of the environment whenever the robot encountered change. The second was a *plan repair strategy* optimised to preserve as much of the existing plan as possible and replan only the changed regions.

As these proposed strategies do not seek to alter the underlying processes of the *offline sampling-based coverage planner*, it is therefore important to understand how the *coverage sampling problem* (CSP) and *multi-goal planning problem* (MPP) algorithms will perform when planning over a submarine tank environment. Therefore, irrespective of the replanning strategy implemented, both approaches are directly reliant on the performance of the existing offline planning algorithms. To the author's knowledge, as there were no comparable research results available on the performance of the *offline sampling-based coverage planner with respect to confined space submarine tank inspection*, the primary focus of this chapter is to investigate the intrinsic functionality of the *offline sampling-based coverage planner* to determine which of the two replanning strategies is better suited to solve the online planning problem. This chapter documents an experiment designed to provide data to assist in making this decision.

The objectives of the experiment were as follows:

Objective 1) Benchmark the performance of the *offline sampling-based coverage planner* within a representative submarine tank environment.

This provided insight into how the coverage planner behaves in different planning environments.

Objective 2) Determine to what extent the *lazy point-to-point planner* (LPP) influences the planning times within the target environment.

In Section 3.4.4, the LPP was listed as a concern due to its long computation times. Understanding the impact of the LPP when it solves the submarine tank planning provides insight into how to reduce planning times.

Objective 3) Discover any other limitations of the coverage planner that may impact online performance.

The *offline sampling-based coverage planner* is applied to four controlled environments to assess its performance against controlled variations within the environments.

4.2 Implementation of the Existing Offline Sampling-based Coverage Planner

The *offline sampling-based coverage planner* was implemented according to the specifications presented in Englot and Hover (2013). However, in preparation for the new planning constraints (Section 3.6), testing environments (Section 4.3) and online implementation, changes were made to particular open source packages that were used in the original implementation (Englot and Hover, 2013).

Amendments were also made to the *offline coverage planner* to handle difficult situations that arose when developing the existing algorithm. When these problems were not handled, it resulted in the coverage planner taking copious amounts of time to complete its intended task, if at all. As these situations would undoubtably appear when executing the online variant of the *offline coverage planner*, it was imperative that the CSP and MPP procedures contained additional algorithms to capture and handle these situations, so they do not impact real-world execution and performance.

The amendments listed in this section focus on how these algorithms capture these difficult situations when generating plans offline. Chapter 7 will discuss how these algorithms are amended again for online use. These changes did not alter the original algorithmic properties

of the CSP and MPP but enhanced their ability to perform reliably across a variety of planning problems. Some of these additions may have been implemented in the existing version of the coverage planner presented in Englot and Hover (2013), however, to the author's knowledge were not discussed. For clarity, this section explains how this recreated implementation handles;

1) the sampling of viewing configurations to create a *redundant roadmap* given the new visibility constraints discussed in Section 3.6.3,
2) the detection and removal of *trapped configurations* due to *prison cell* geometries,
3) how *unobservable primitives* are removed from the planning problem to avoid being continually sampled and recorded to notify the operator during operation,
4) changes made to the LPP which included, the substitution of a Travelling Salesman Problem (TSP) solver to accommodate larger covering sets, and
5) handling point-to-point paths that exceed the allocated solution time within the LPP.

The recreated *offline sampling-based coverage planner* used in the experiment in this chapter creates the baseline planner that is the foundation of the *online adaptive coverage sampling-based coverage planner* that is discussed in following chapters. A summary of the software packages used to replicate the original implementation is specified in Appendix A.

4.2.1 Constructing a Redundant Roadmap

Sampling the viewing configurations to construct a *redundant roadmap* is procedurally undertaken as described by Englot and Hover (2013) (Section 3.3.1; Algorithm 3-1). To summarise, creating a *redundant roadmap* requires the following steps:

1) Randomly select a primitive that has not met the required sightings (*redundancy*).
2) Randomly sample a configuration within the neighbourhood of the selected primitive.
3) Determine that the configuration is collision-free and can be reached by other configurations. Section 4.2.2 elaborates further on determining the reachability of a configuration.
4) Calculate coverage of the configuration over the surface.
5) If the configuration sights at least one primitive that is required to be sighted, add the configuration to the *redundant roadmap*.
6) Repeat steps 1-5 until all primitives have met redundancy or have been removed from the planning problem (Section 4.2.3).

For this implementation, configurations are still randomly sampled within a *spherical coordinate system*, r (range), θ (polar angle), ϕ (azimuth) before being transformed to the world frame based on the normal of the selected primitive (Figure 4-1). Details to this rotating from the spherical frame to cartesian frame can be found in (Sadiku, 2014). The sampling ranges for r, θ and ϕ are defined in Equations 4-1 to 4-3 based on the minimum and maximum Field of Depth (FOD_{min} and FOD_{max}) and Field of View (FOV_{angle}).

$$r: FOD_{min} < r < FOD_{max} \quad (4\text{-}1)$$

$$\theta : 0 < \theta < 2\pi \quad (4\text{-}2)$$

$$\phi : 0 < \phi < FOV_{angle} \text{ (rads)} \quad (4\text{-}3)$$

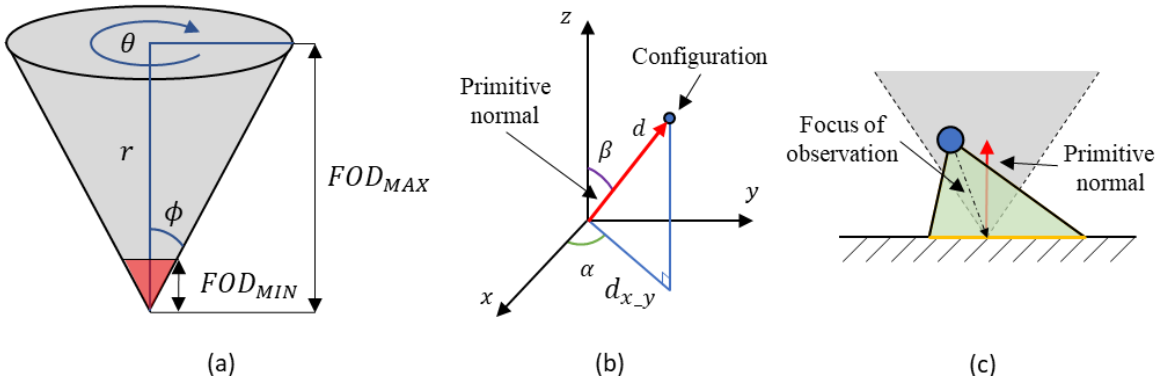


Figure 4-1: Transforming a configuration sampled in the spherical frame to the world frame. (a) A configuration is sampled in the spherical frame based on the constraints of the sensor. (b) Transformation of the configuration to the world frame can be achieved by rotating and offsetting the configuration around a primitive normal in the world frame. (c) The result of the transformation angles the configuration to the centre of the selected primitive.

For static collision checking, OPCODE (*Optimal Collision Detection*; Terdiman, 2003) replaces PQP (*Proximity Query Package*; Larsen et al., 2000). As the static collision check is performed over a spherical robot model (Section 3.6.2), the collision check made between the environment model and the robot model is simple and has minimal degradation in performance to PQP (Trenkel, Weller, and Zachmann, 2007).

Capturing the local neighbourhood of primitives surrounding the configuration is achieved by using FLANN (*Fast Library for Approximate Nearest Neighbours*; Muja and Lowe, 2009) implemented through the PCL (*Point Cloud Library*; Rusu and Cousins, 2011) as implemented previously (Figure 4-2). FOD_{max} was used as to limit the radial nearest neighbour search (Figure 4-2a).

It was found that creating a nearest neighbour search tree using the Euclidean coordinates of the incentre of the triangle, rather than the vertices of the triangle, resulted in significantly

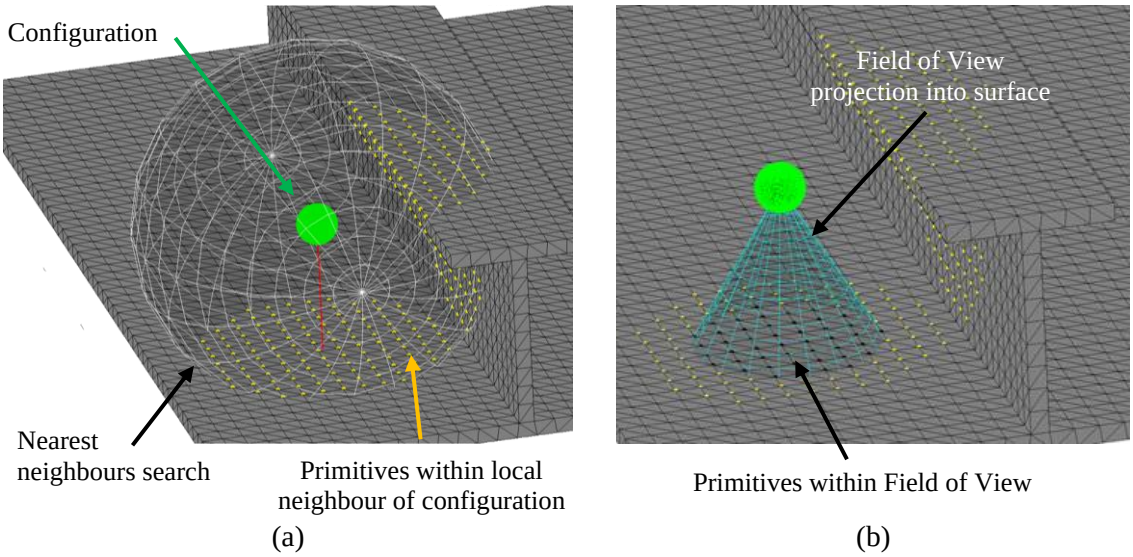


Figure 4-2: Visualisation of how the visibility checks were undertaken. (a) A radial nearest neighbour search finds the primitives within the neighbourhood of the configuration based on the maximum Field of Depth (FOD) specified. (b) Primitives within the local neighbourhood are pruned to the Field of View (FOV) of the sensor.

faster search results[2]. As there is only one incentre per triangle, the indices returned by the search corresponded to a specific triangle. Creating a search tree of vertices required additional checks to ensure that all three vertices were within the neighbourhood to determine if a triangle was within the FOV. These additional checks increased planning times significantly. The images in Figure 4-2 shows the earlier implementation of visibility checks using a nearest neighbour search over the vertices.

The coverage was calculated by determining the angle of each primitive's normal vector ρ against the vector of the sensor's projection γ (Figure 4-3). If the resulting angle ξ between ρ and γ was less than ϕ, it was considered to lie within the FOV and was checked for occlusion using a ray trace.

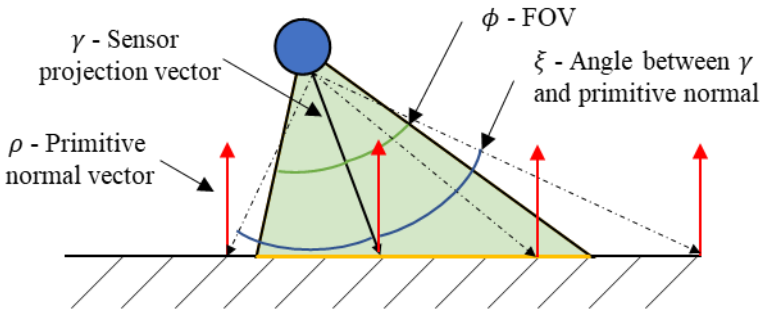


Figure 4-3: Calculation of a configuration's coverage.

[2] Improvement courtesy of Lee Ying Wu

Ray tracing was only cast to the incentre of the primitive. A ray trace to each vertex of the primitive would be a more thorough approach but it would result in more ray traces. Given the resolution of the models was 100mm, a single cast was sufficient to determine occlusion. *OpenSceneGraph* (Wang and Qian, 2010) was retained to perform ray tracing.

4.2.2 Handling Trapped Configurations due to *Prison Cells*

When sampling configurations within tight enclosed spaces, there is a possibility that a configuration can be generated within an area of the environment that would not allow any feasible path the ability to connect the configuration to the plan. This problem has been described as the *prison cell* problem (Englot and Hover, 2012a) and these configurations need to be removed from the planning before solving the MPP. To overcome the *prison cells*, Englot and Hover (2012a, 2013) suggests connecting all configurations to a common origin. In this work an alternative approach to determining configurations trapped due to *prison cells* is applied.

In an environment where all objects are known to be fully enclosed, configurations trapped within *prison cells* are avoided by performing an additional check procedure during the sampling phase. This procedure casts a ray from the position of the configuration through to the environment's exterior boundary. Counting the number of intersections that each ray-trace makes with each object, through to the exterior boundary, determines whether a configuration is inside or outside a *prison cell* geometry (Figure 4-4). The number of intersections to determine if a configuration exists in free space is dependent upon the layers of the exterior boundary mesh manifold. Configurations in free space inside a double outer-layer mesh manifold will return an even intersection count. For single-layer mesh manifolds the logic is reversed. This procedure is known as the *even-odd algorithm* (Shimrat, 1962; Haines, 1994) but is termed the *Trapped Configuration Heuristic* (TCH) in this thesis.

As all the environments are two-layer manifolds with all objects within the environment constructed as single layer (no interior surfaces), an even intersection constraint is placed on four casted rays traced to the exterior boundary of the environment in the $\pm x$ and $\pm y$ directions. For a configuration to be considered valid within free space, all four traces must return an even count.

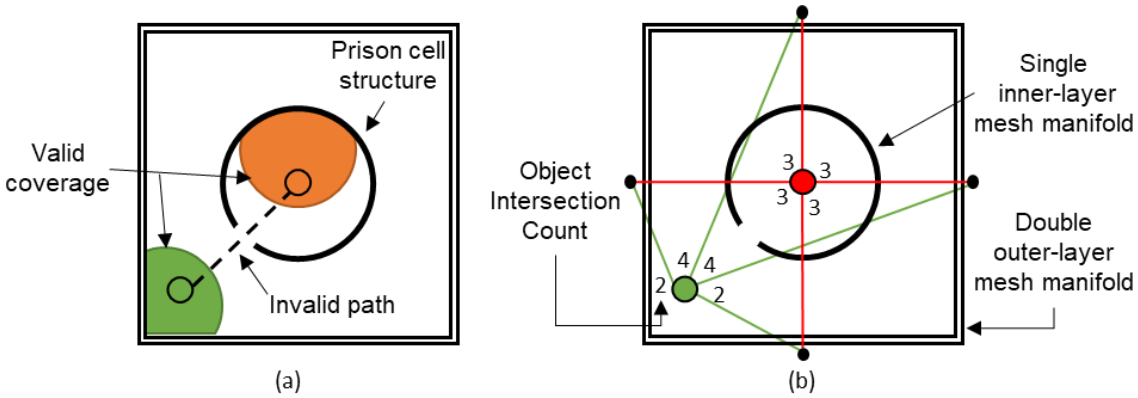


Figure 4-4: Identifying trapped configurations within *prison cells*. (a) A basic example of a prison cell geometry trapping a configuration inside an object prohibiting a feasible path to other configurations. (b) The number of intersections each ray trace makes with interior objects and the mesh boundary determines if a configuration is within free space.

4.2.3 Introducing a Primitive Rejection Limit to Avoid Over-Sampling Unobservable Primitives

In Section 3.4.3, an assertion was formed from a raised concern that it would be acceptable to relax the 100% coverage constraint providing the areas that cannot be feasibly covered are recorded and reported to the operator. *Prison cells* or *complex geometries* have the potential to create a situation where primitives contained within or around these geometries can never be observed. The robot may not physically be able to fit within these certain areas of the environment or structures may completely occlude other structures. As these *unobservable primitives* are unlikely to reach the desired redundancy, a new user parameter, the *primitive rejection limit*, was added to avoid continually sampling these primitives if no observations are likely to be obtained. Given the *offline coverage planner* is aware of how many primitives are in the planning problem, keeping a record of any primitives unable to meet the required redundancy is trivial.

The *primitive rejection limit* assigns a limit for the number of failed attempts a primitive receives before it is deemed *unobservable*. Once a primitive has been deemed *unobservable* it is removed from the *unobserved primitives list* (Section 3.3.1), added to the *unobservable primitive list* and will no longer be a candidate for sampling for the remainder of the sampling process. An *unobservable primitive* can only be removed from the *unobservable primitive list* if it is indirectly observed by another configuration, regardless of the *unobservable primitive's* redundancy count. This ensures that all observed primitives are represented in the final coverage plan.

At the conclusion of the sampling process, the *unobservable primitive list* can be used to inform the operator of the unobservable regions within the environment where the robot was

unable to acquire coverage. The *primitive rejection limit* was set at 20 as this number had been determined to be a good balance between oversampling an *unobservable primitive* and generating too few samples that falsely reported the primitive as *unobservable*.

4.2.4 Substitution of the TSP Solver for Large Covering Sets

In anticipation that the constraints placed on the planning problems will produce larger covering sets to the set sizes presented by Englot (2012) and Englot and Hover (2013), a substitution of the TSP solver was made. The *Christofides algorithm* (Christofides, 1976) was substituted with the *Quick-Borůvka* algorithm (Applegate et al., 2003) to provide the *Chained Lin-Kernighan algorithm* (CLK) with an initial TSP solution. In this thesis this pairing is referred to as QB-CLK (*Quick-Borůvka CLK*). For smaller problems size less than 10,000 cities, the QB-CLK produces comparable solutions in a shorter time to that of a *Christofides-CLK* pairing (Applegate et al., 2003). As the LPP is an iterative planner, any effort to reduce the time to calculate a TSP solution, per iteration, is a benefit.

This implementation utilises *Concorde's* implementation of the QB-CLK TSP (Applegate et al., 2003). Englot and Hover (2012b) made a similar change when the *Christofides algorithm* is replaced with *Concorde's Nearest Neighbours TSP solver*. No reason was supplied or reports of computational deficiencies where mentioned about this change.

With the expectation of large covering set sizes, the CLK is given two solution times to solve the TSP. For covering set sizes of under 600, 0.5 seconds was enough time to find a solution without timing out. Covering set sizes greater than this threshold were given one second. In practice, smaller set sizes solved quicker than 0.5 seconds and did not require the full allocated time before terminating with a solution. The change in threshold allowed larger sets more time to stabilise to an answer as the CLK was run once per iteration. All other parameters to initialise QB-CLK TSP were left as their defaults.

4.2.5 Motion Planning and Handling Unachievable Paths

The *RRT-Connect algorithm* (Kuffner and LaValle, 1998) remains as the designated point-to-point path planner. To prevent computing an RRT path for every chosen edge, a direct path between configurations is evaluated first. If the direct path fails to create a valid path, an RRT is called to generate a solution. OMPL (*Open Motion Planning Library*; Sucan, Moll and Kavraki, 2012).

In reality, solving the path for a robot of high-DOF every path would be resolved with an

RRT. However, in the case of a minimally constrained robot, such as a robot containing 6-DOF with holonomic constraints, direct point-to-point path evaluations checks are acceptable. Each RRT is given up to 0.5 seconds to solve for a path.

In the unlikely case a valid path cannot be found in the allotted time, the edge between the two configurations is still labelled as an *uncleared edge* (Algorithm 3-2). To avoid the likelihood TSP solver selecting the same edge again in the next iteration the initial estimate of the distance is doubled.

4.3 Planning Scenarios to Evaluate Algorithmic Properties

Generating offline plans directly over the representative submarine tank environments only provide an insight into how the algorithms work in one particular scenario. Since submarine tanks are considered to be complex environments, it is a challenging problem to determine if the effects on the *offline coverage planner* are due to environmental influences or the algorithmic process. Therefore, to help generalise the functionality of the *offline sampling-based coverage planner*, a set of simpler environments were introduced.

The planning problems presented have been designed to explore various aspects of the *offline sampling-based coverage planner*. The planning problems used in this chapter include two empty box environments (*2x2m* and *6x6m*), two simplified house plans (*House* and *House-W*) and two representative submarine tanks (*Tank* and *Tank-P4*). Each planning set comprised a base model that was simpler to solve than that of the paired model.

Collectively, these planning environments are designed to control properties such as;

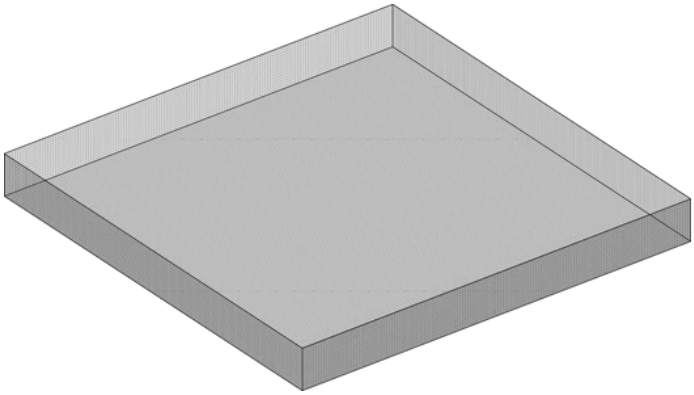
1) the number of primitives, or
2) the geometry of the space.

Controlling these properties enables each model set to evaluate a particular algorithmic process. Besides the two submarine tank models, which provide an insight into the performance of the *offline sampling-based coverage planner* inside the target environment, the additional models examined how the sampling and path planning function under controlled geometrical changes.

For all planning environments presented, the coverage planner was only required to provide coverage to the interior surfaces of the environment. The maximum resolution of primitives in these mesh models was 100mm.

4.3.1 Empty Box Environments

The two empty box examples *2x2m* and *6x6m* (Figure 4-5) were used to demonstrate the influence of the sampling procedures to a change in environment size when no extensive planning was required. It was expected that the planning times would be dominated by the time taken to solve the CSP.



2x2m	
Dimensions	2.0x2.0x0.2m
Primitives	58,058

6x6m	
Dimensions	6.0x6.0x0.2m
Primitives	41,0432

Figure 4-5: Planning environments *2x2m* with *6x6m*.

With no obstacles present within the environment, the LPP was expected to take no more than two planning iterations to provide a solution as no RRTs were required to update any paths, given that all configurations are reachable via a direct path to one another. This ensures that the minimal amount of effort was required by the LPP to produce a solution. The only expected difference between the time it takes to solve the MPP between *2x2m* and *6x6m,* would be due to size of the each covering set.

4.3.2 House Plans Containing a Single Geometrical Change

Contained within the same 2.0 x 2.0m box as *2x2m,* two-similar scaled-down house plans, *House* and *House-W* introduce further controlled complexity to the problem set (Figure 4-6). The walls of the house plans introduced more opportunity for collision and the varying routes throughout the environment make path planning a little more challenging. Designed to have approximately the same configuration density to control the time it took to solve the CSP, the differentiating factor is the inclusion of an additional wall to *House-W* on the exterior border that is intended to impact the LPP. As it was unknown when or where new changes would occur and the impact these changes would have on the coverage planner, understanding the response of the LPP to small changes in the environment was important for online planning.

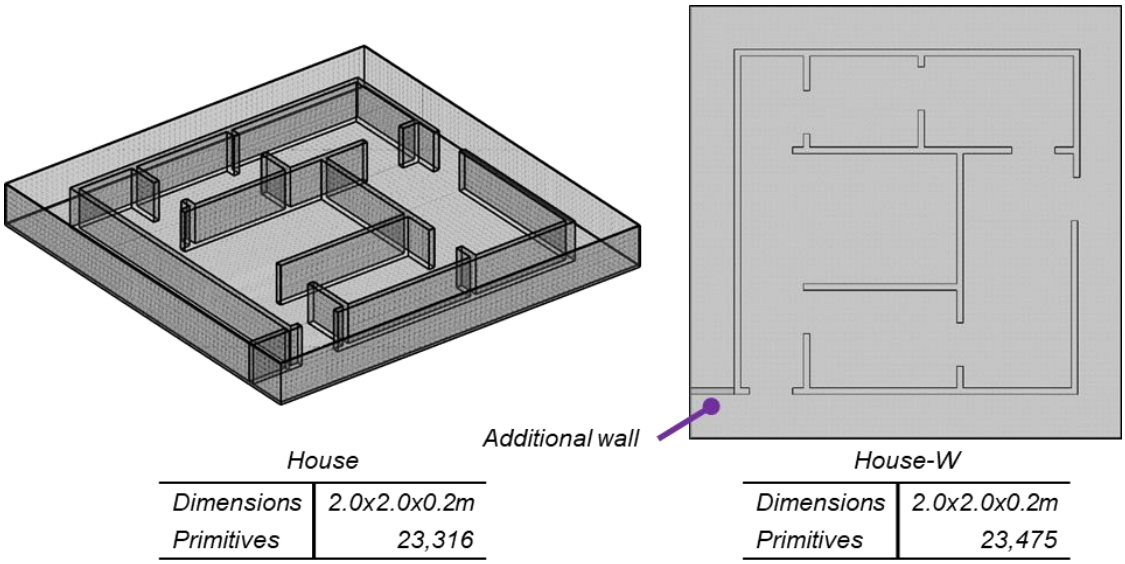


Figure 4-6: Planning environments *House* and *House-W*.

While the additional wall is only a small geometric inclusion with respect the size of the environment, it blocks entry into the second doorway and therefore will invalidated several path options that were available for *House*. As a result, the *Euclidean assumption* that was used to estimate the initial connectivity between all configurations, was significantly underestimated around the change for *House-W* than it was for *House*, resulting in more RRTs to rectify the initial estimations. Consequently, planning times for *House-W* were expected to be longer than *House*.

4.3.3 Representative Submarine Tank Environments

The representative submarine tank environments, *Tank* and *Tank-P4* (Figure 4-7) were designed to provide a baseline understanding of how the *offline sampling-based coverage planner* solves the target application. These tank models were synthetically designed to represent and contain features akin to those expected in real-world models given that presentation of and testing on real-world submarine tank data was prohibited (Section 1.5.2). These models were designed to focus on coverage planning within smaller tank environments.

Planning within a smaller tank environment posed a difficult problem to solve due to the confined spaces between structures. Unlike larger, more open spaces, the tight spaces reduced the sampling areas in which maximal coverage, or any coverage, of a surface can be obtained. Figure 4-8 shows three different examples of how the valid sampling region used to generate viewing locations within the local neighbourhood changes around an I-beam. When sampling around complex geometries, several viewing configurations may be

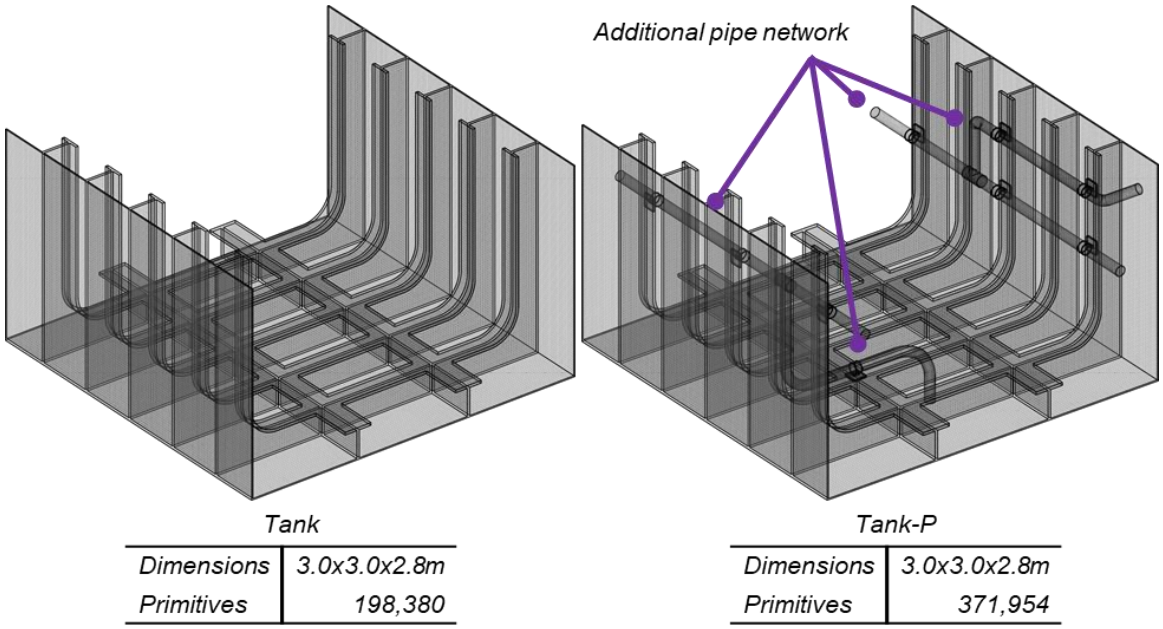


Figure 4-7: Planning environments *Tank* and *Tank-P4*.

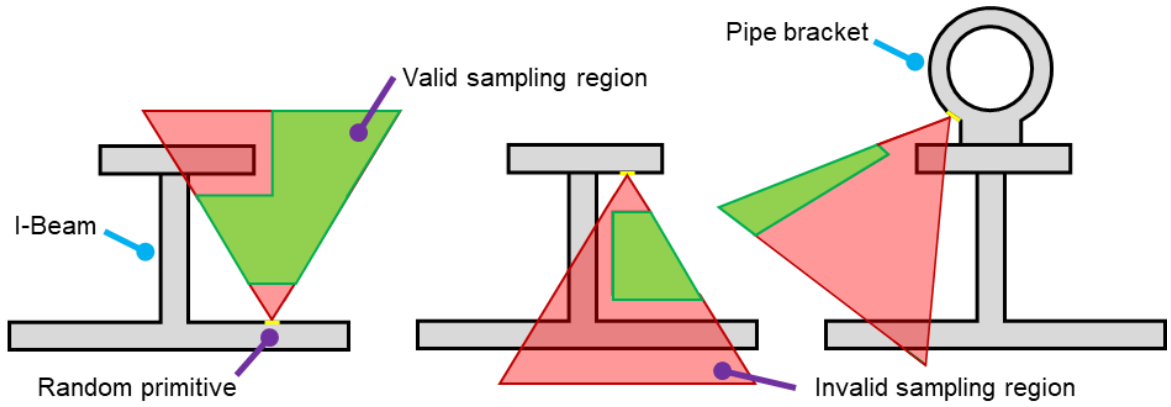


Figure 4-8: Sampling configurations around complex geometries such as I-beams and pipe brackets can be challenging in clustered areas.

rejected due to collision or visibility constraints. In some cases, visibility to a particular primitive can be quite restrictive, like the underside of a pipe bracket (Figure 4-8c), and may require several viewing positions to cover a small area of a structure.

The base model *Tank* contains a basic tank layout of repeating I-beam structures in both the horizontal and vertical directions. *Tank* has dimensions of 3.0m x 2.8m x 3.0m to represent a tightly compact and confined environment between I-beam structures. *Tank-P4* increased the complexity of the problem by overlaying a pipe network over the sides of the base environment. *Tank-P4* was expected to be the most difficult environment to solve of the planning set; the addition of pipes and pipe brackets creates areas of higher primitive occlusion resulting in greater covering-set sizes than *Tank* which also added to the increased complexity of generating a plan. Given the visibility constrains placed on the planning

problem (Section 3.6.3), it was expected that covering sets for *Tank* and *Tank-P* would be in the thousands.

4.4 Benchmarking the Offline Sampling-based Coverage Planner

A Monte Carlo experiment was run over 20 random primitive sets to investigate the functionality of the *offline sampling-based coverage planner*. To test for any variability within each primitive set, each configuration set was trialled five times (n = 100). The 20 different sets allow a direct comparison over different coverage distributions while the five trials determine the variability of planning times and tour lengths in response to an identical primitive set.

To ensure the comparisons within each primitive set are valid, each of the five trials start and finish their tours at the same location. The starting position for each planning problem is assigned to the configuration with the highest number of primitives. This location is determined by the first configuration selected when solving the *set cover problem* (SCP). For each of the five trials in each random primitive set this location will always be the same. To avoid the LPP taking an excessive time to solve a particular MPP, a six-hour time limit was imposed on the LPP. If any planning scenario reached the time limit, the answer from the current iteration of the LPP is provided as the final answer.

For each planning scenario a feasible inspection plan is generated from a *redundancy-ten roadmap* using the simplified constraints presented in Section 3.6, Table 3-1. In summary, the chosen FOV was ±45° with a sensing range between 100-to-270mm (FOD_{min}, FOD_{max}). The robot collision sphere encapsulating the end effector has a diameter size of 50mm.

The *offline sampling-based planning planner* with amendments (Section 4.2) was implemented in C++ and implemented on a 64-bit Intel i7 920 CPU with 8 cores at 2.67GHz with 6GB RAM running Ubuntu 16.04 LTS. All data analysis was performed using MATLAB 2018b.

4.4.1 Computational Observations and Results

Table 4-1 shows the statistical analysis of the overall planning times, covering set sizes and final tour lengths for each of the planning scenarios. Table 4-2 provides the computational time it took to solve the CSP and the MPP. The CSP time reflects the average time taken to sample the *redundancy-ten roadmaps* and to solve the SCP. The MPP time reflects the time

Table 4-1: Overall planning times, configuration set sizes and overall tour lengths for each of the planning scenarios.

Model	*Trials*	*Timeouts*	*Overall Time (s)*				*Configurations*				*Tour Length (m)*			
			$\bar{x}$	*SD*	*Median*	*IQR*	$\bar{x}$	*SD*	*Median*	*IQR*	$\bar{x}$	*SD*	$\bar{x}$	*IQR*
2x2m	*100*	*0*	*2.88*	*0.31*	*2.79*	*0.25*	*152.00*	*2.96*	*152.00*	*5.50*	*21.47*	*0.27*	*21.49*	*0.53*
6x6m	*100*	*2*	*2,704.35*	*4,246.76*	*1,049.51*	*2,569.38*	*1,011.05*	*8.07*	*1,013.00*	*13.00*	*159.46*	*1.07*	*159.74*	*1.64*
House	*100*	*0*	*14.09*	*7.39*	*13.40*	*10.23*	*321.80*	*5.42*	*321.00*	*6.50*	*33.94*	*0.56*	*33.93*	*0.87*
House-W	*100*	*0*	*7,692.39*	*1,616.25*	*7,938.56*	*2,900.23*	*325.35*	*4.90*	*324.00*	*7.00*	*35.72*	*0.46*	*35.68*	*0.60*
Tank	*100*	*14*	*6,581.49*	*7,274.07*	*2,986.16*	*8,996.53*	*1,301.76*	*7.23*	*1,305.00*	*12.00*	*148.99*	*1.08*	*148.86*	*1.15*
*Tank-P4**	*50*	*45*	*20,984.90*	*2,659.19*	*21,666.60*	*1.40*	*1,654.70*	*6.79*	*1,653.50*	*10.00*	*175.65*	*1.26*	*175.46*	*1.35*

**Only 5 trials completed within the six-hour time limit*
IQR – Interquartile Range

Table 4-2: *Coverage sampling problem* (CSP) and *multi-goal planning problem* (MPP) planning times for each of the planning scenarios.

Model	*CSP Time (s)*				*MPP Time (s)*			
	$\bar{x}$	*SD*	*Median*	*IQR*	$\bar{x}$	*SD*	*Median*	*IQR*
2x2m	*2.55*	*0.05*	*2.55*	*0.06*	*0.32*	*0.30*	*0.23*	*0.24*
6x6m	*21.89*	*0.69*	*21.97*	*0.67*	*2,682.37*	*4,246.79*	*1,027.67*	*2,569.86*
House	*2.60*	*0.03*	*2.60*	*0.03*	*11.48*	*7.40*	*10.80*	*10.21*
House-W	*2.64*	*0.03*	*2.64*	*0.03*	*7,689.72*	*1,616.24*	*7,935.92*	*2,900.27*
Tank	*32.53*	*0.39*	*32.63*	*0.46*	*5,898.72*	*6,896.53*	*2,739.26*	*6,521.80*
*Tank-P4**	*65.61*	*0.85*	*65.64*	*0.92*	*20,919.20*	*2,659.04*	*21,600.80*	*0.60*

** Averaged of 50 trials*
IQR – Interquartile Range

taken for the LPP to generate a solution. Therefore, it includes the time taken to create the initial adjacency matrix, iteratively solving the TSP and generated paths using the *RRT-Connect algorithm*. The coverage plans for each planning problem pair can be seen in Figures 4-9 to 4-11. All planning problems achieved 100% coverage of the internal surfaces and no primitive exceeded the *primitive rejection limit*.

Given all the results, the first observation made is that there was a clear distinction between the smaller and larger planning problems. The *smaller planning problems* (*2x2m*, *House* and *House-W*) all had configuration set sizes no greater than 400 configurations. The *larger planning problems* (*6x6m*, *Tank* and *Tank-P4*) had covering sets into the thousands.

In comparison to the computational results published by Englot and Hover (2013), only the covering sets of the *smaller planning problems* are equivalent to those generated from *redundancy-ten roadmaps* in Englot's work. The *larger planning problems* contained covering sets of up to three to eight times greater than those previously published. These results highlight that the *offline sampling-based coverage planner* is capable of generating coverage plans of significant size, but it has come at the expense of excessive computational times.

While a direct comparison cannot be made between these results and the computational results of Englot and Hover (2013) due to the significant differences between planning environments and planning constraints, the results of Englot (2012) and Englot and Hover (2013) were used as a guide to predict the outcomes of these experiments. As a benchmark for the results in this experiment, the planning times reported for Englot's largest planning problem, the *USCGA Seneca*, took no longer than 19 minutes to construct a full plan (Englot and Hover, 2013). The resultant coverage plan consisted of over 400 configurations over the 70m structure. However, given that the environment sizes in this experiment were significantly smaller, it was not expected that any of the planning problems, especially those of similar covering set sizes, would exhibit planning times significantly larger than the *USCGA Seneca* example.

Table 4-1 clearly shows that only *2x2m* and *House* were solved in under 19 minutes. The other planning problems, *6x6m*, *House-W*, *Tank* and *Tank-P* all exhibited planning times that exceeded 45 minutes. Plans for *House-W* executed in just over 1.5 hours and for *Tank* just over two hours. The worst performing scenario was *Tank-P* which averaged a time of six hours. Given that 90% of *Tank-P4's* 50 trials failed within six-hours it was enough to determine that 50 more trials would not produce a better outcome. While it was expected that *Tank* and *Tank-P* would generate large covering set sizes, there was no expectation that solving these problems

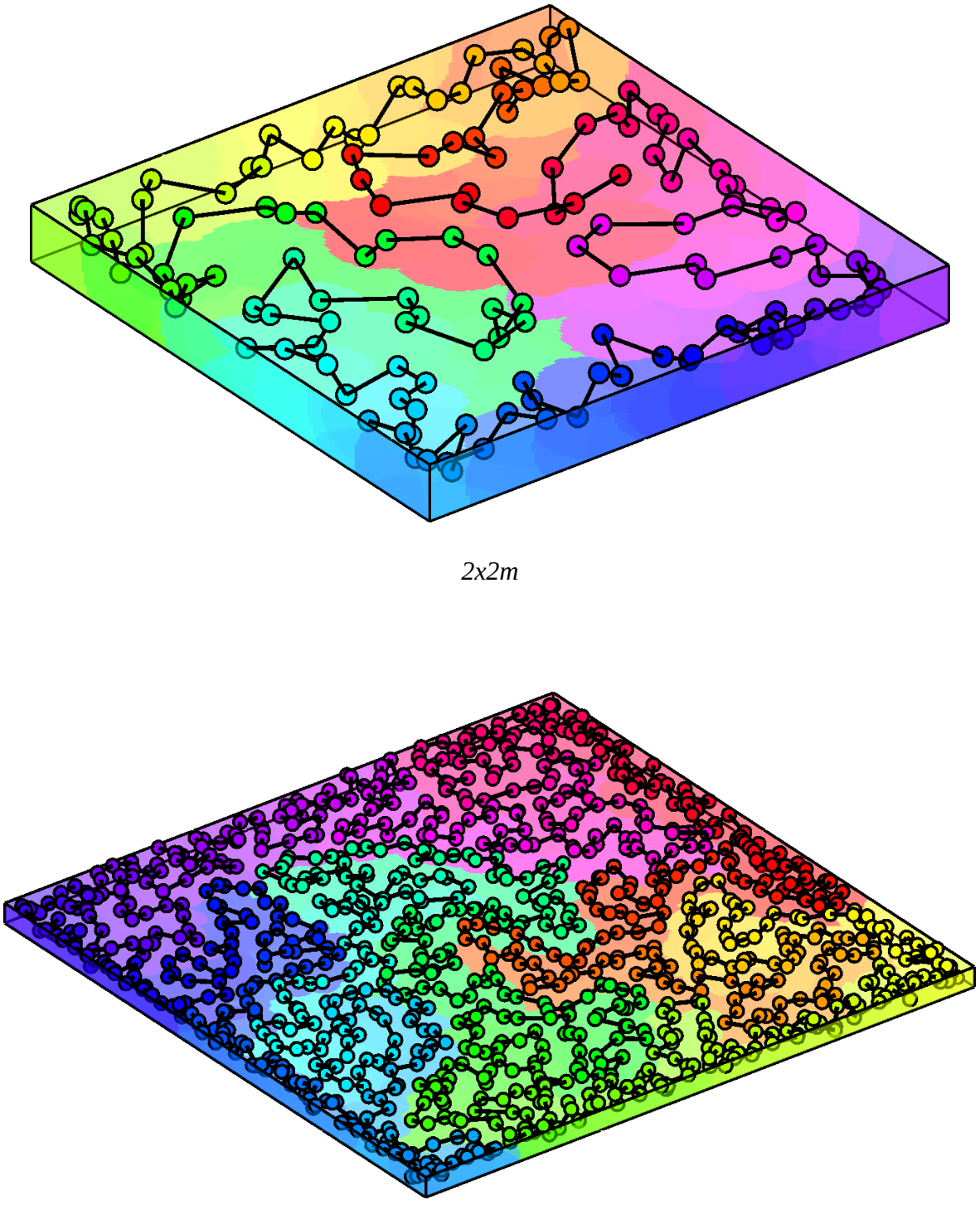
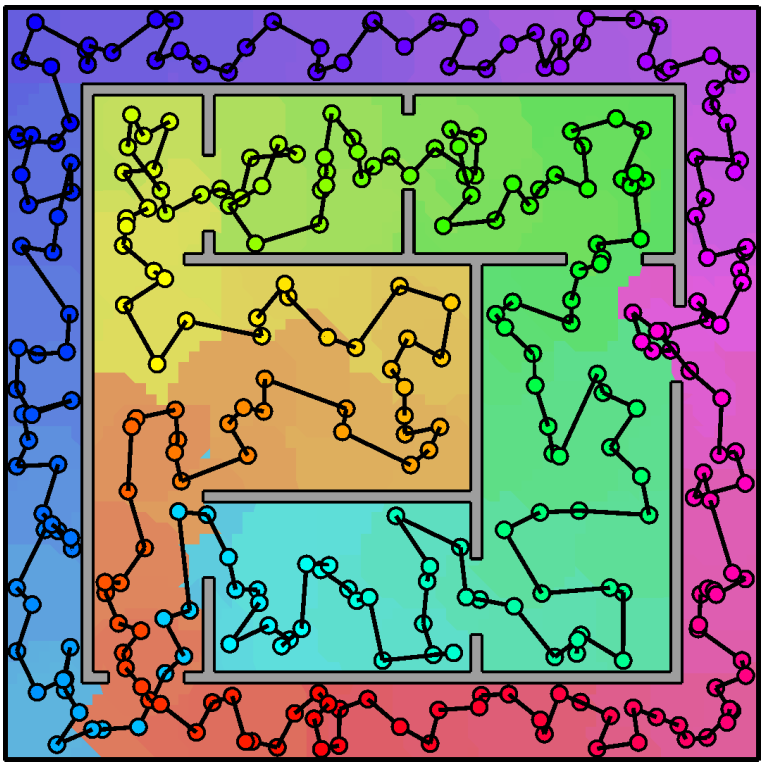
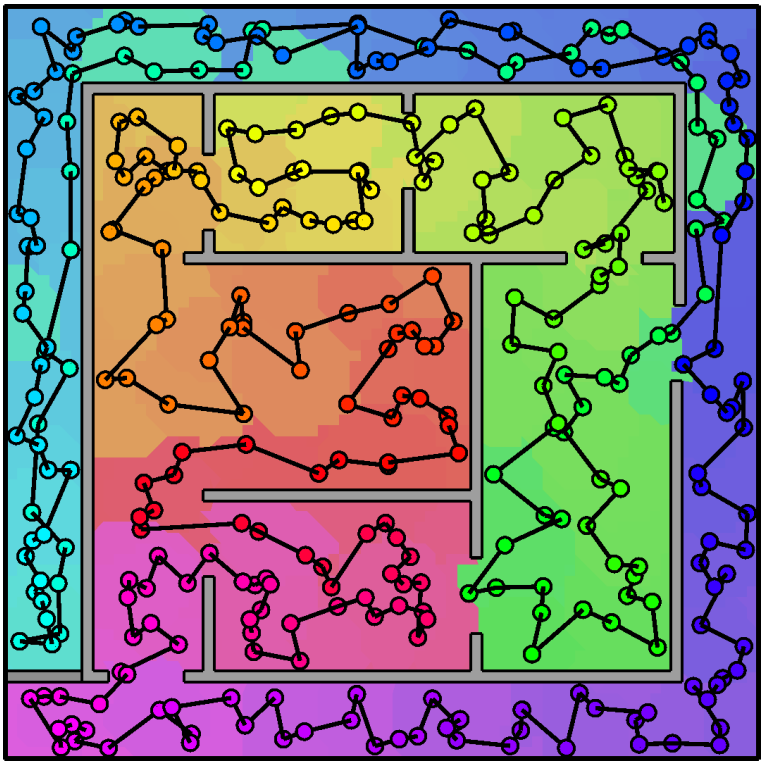


Figure 4-9: Generated coverage plans for *2x2m* and *6x6m*.



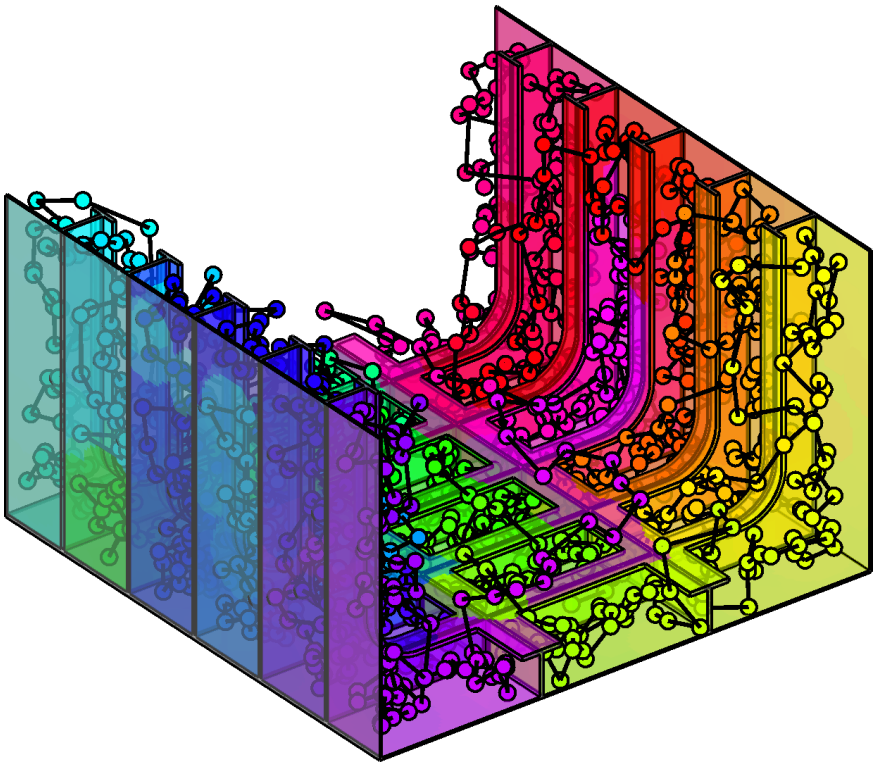
House



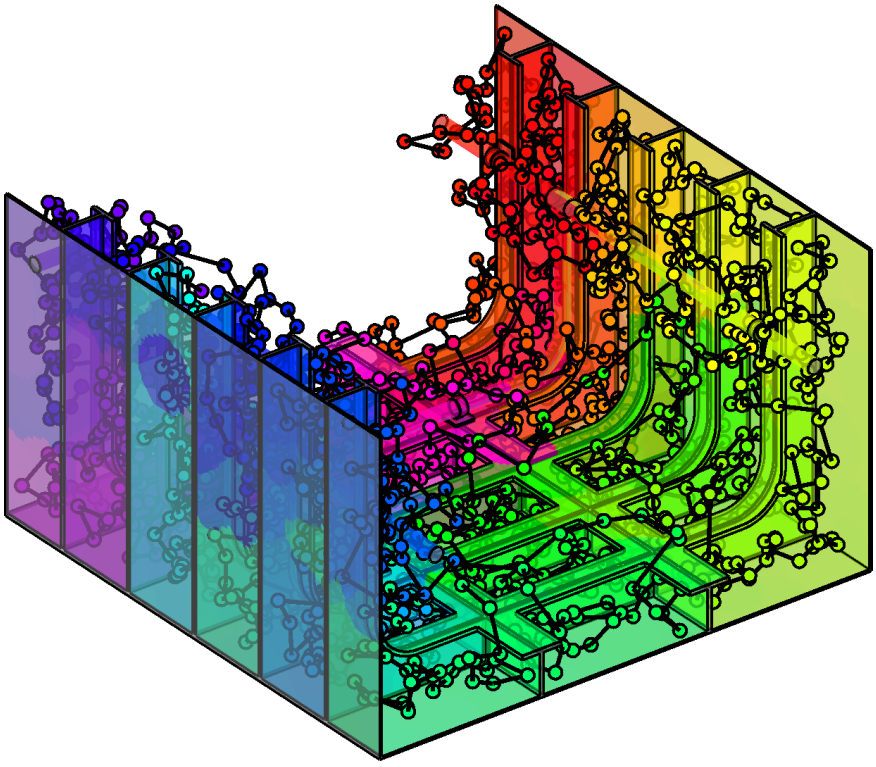
House-W

Direction of Coverage Plan

Figure 4-10: Generated coverage plans for *House* and *House-W*.



Tank



Tank-P4

Direction of Coverage Plan

Figure 4-11: Generated coverage plans for *Tank* and *Tank-P4*.

with simplified planning constraints would ever reach the six-hour time limit. The six-hour time limit was implemented as a practical limitation to stop the simulation from executing too long if a problem occurred. If these results were taken as the baseline for an online planner, running replanning updates that take six-hours is not at all feasible.

The main concern expressed about the *offline sampling-based coverage planner* was large computational costs associated with generating coverage plans, in particular the solution times of the LPP (Section 3.4.4). An examination of the MPP times in Table 4-2 confirm this concern. Besides *2x2m*, which solving the MPP accounted for only 11% of the overall time, *6x6m*, *House*, *House-W*, *Tank* and *Tank-P4* all had over 81% to 99% of their computational time dedicated to solving the MPP.

A closer examination of the MPP times revealed that there was *a significant variability present in the LPP that has negatively impacted overall planning times across all planning scenarios.* The spread of the variation is captured by the median and *interquartile range* (IQR) for each planning scenario (Tables 4-1 and 4-2). For the *larger planning problems* there is a significant difference between the means and medians. *6x6m* has a 27-minute difference between the mean and median while just under one hour separates the mean and median for *Tank*. *Tank-P4* has a smaller difference between mean and median due to a majority of trials timing out. Figures 4-12a, 4-13a and 4-14a plot the distribution of overall planning times for *6x6m*, *Tank* and *Tank-P4* and highlight that the distributions are positively skewed.

Table 4-3 shows the *coefficient of variation* of the overall, CSP and MPP planning times, configuration set sizes and resulting tour lengths. These results further illustrate the significant variability the LPP had on producing a solution. The results also show that whilst solving the CSP does not contribute significantly to the overall planning times, the algorithms that solve the CSP do not contribute to variations witnessed within the coverage planner. Given that the same planning constraints were supplied, planning times for the CSP were consistently repeatable despite the random sampling process and the complexity of the environment.

The variation observed across all the trials could have been due to the complexity of solving a particular primitive set. If a particular primitive set was skewing the overall planning times, it would be present across each of the five trials. The overall planning times for the five trials for each of the 20 random primitive sets for *6mx6m*, *Tank* and *Tank-P4* are shown in Figures 4-12b, 4-13b and 4-14b respectively. These figures highlight that the variation in planning times exists between different primitive sets and also within the same primitive set.

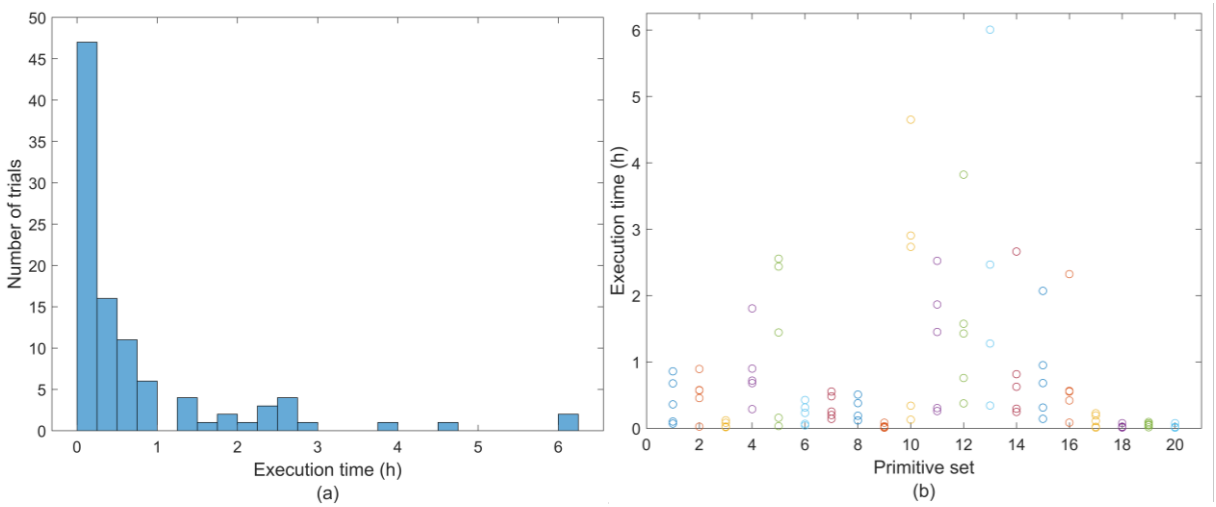


Figure 4-12: Overall planning times for *6x6m*.

(a) Distribution of all planning times. (b) Distribution of planning times within each primitive set.

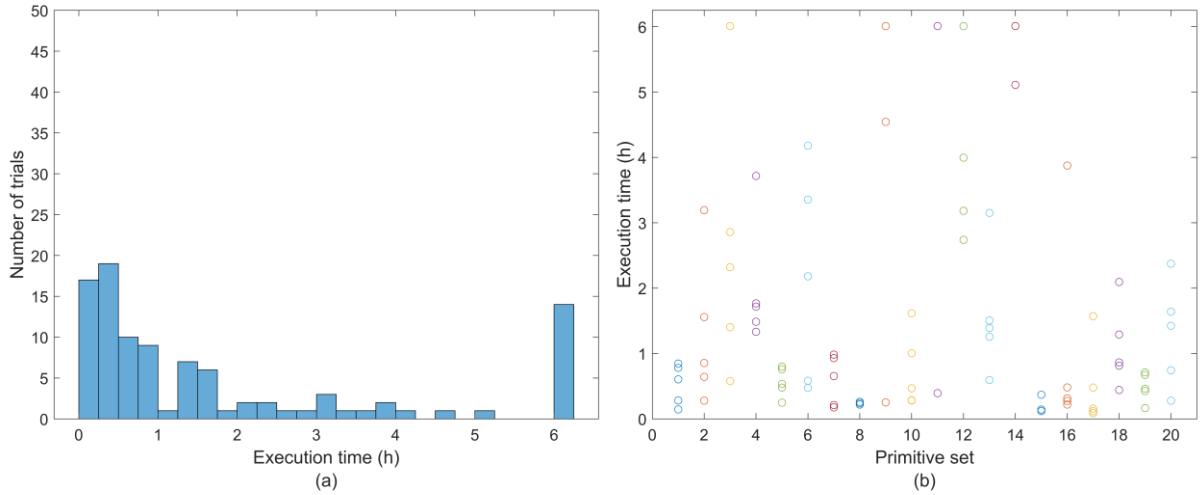


Figure 4-13: Overall planning times for *Tank*.

(a) Distribution of all planning times. (b) Distribution of planning times within each primitive set.

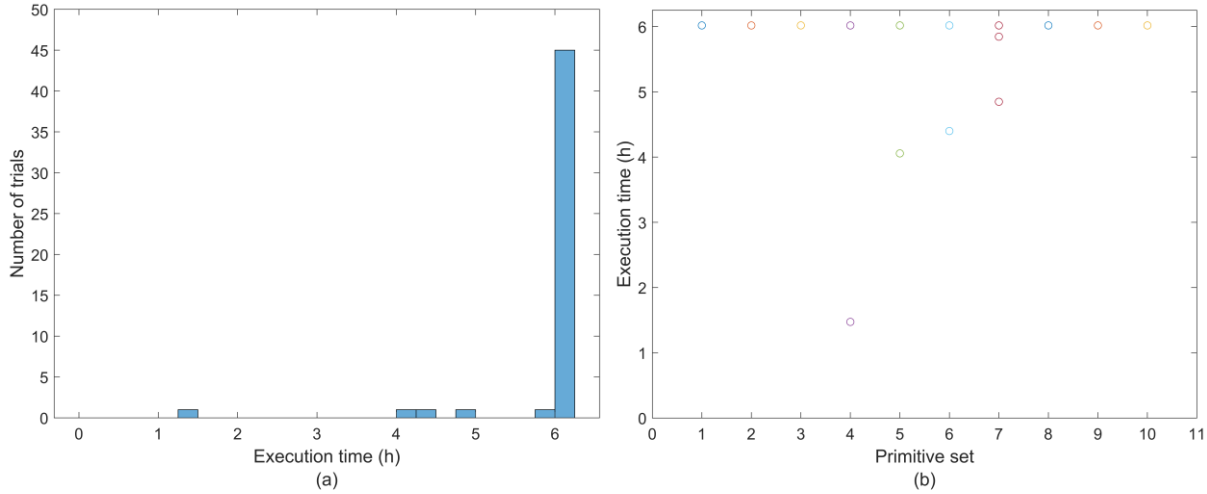


Figure 4-14: Overall planning times for *Tank-P4*.

(a) Distribution of all planning times. (b) Distribution of planning times within each primitive set.

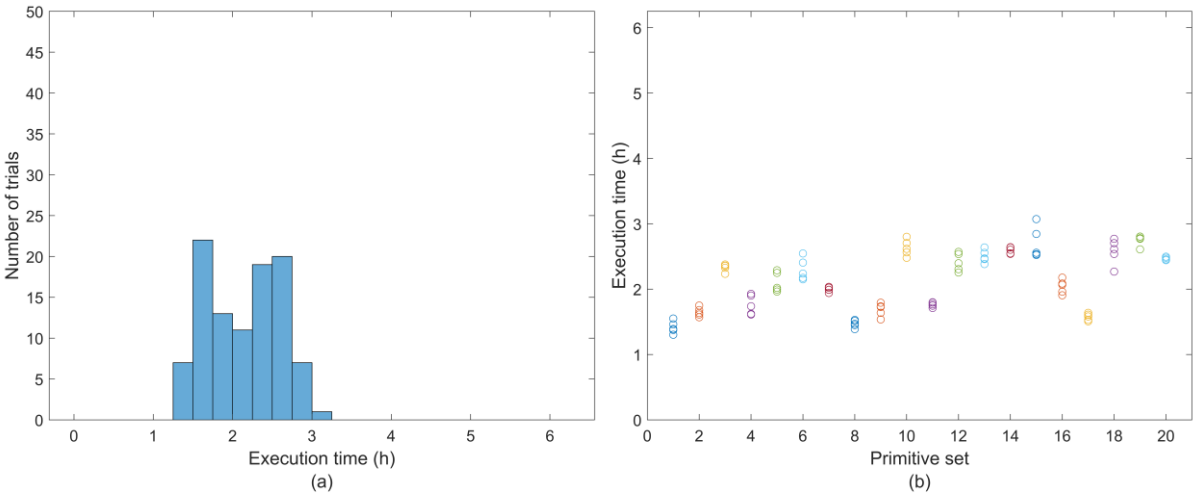


Figure 4-15: Overall planning times for *House-W*.
a) Distribution of all planning times. b) Distribution of planning times within each primitive set.

Table 4-3: Coefficient of variation across the planning data for all planning scenarios.

Model	*Overall Time (%)*	*CSP Time (%)*	*MPP Time (%)*	*Configurations (%)*	*Tour Length (%)*
2x2m	*10.63*	*1.77*	*92.85*	*1.95*	*1.27*
6x6m	*157.03*	*3.13*	*158.32*	*0.80*	*0.67*
House	*52.43*	*1.22*	*64.43*	*1.68*	*1.65*
House-W	*21.01*	*1.22*	*21.02*	*1.50*	*1.28*
Tank	*110.52*	*1.22*	*111.07*	*0.56*	*0.72*
*Tank-P4**	*12.67*	*1.29*	*12.71*	*0.41*	*0.71*
**Tank-P4 has a small variation due the 90% of trials timing out*					

The five trials in Primitive Set 13 for *6x6m* in Figure 4-12b clearly demonstrates the variability of planning times between the trials of a primitive set. Primitive Set 13 contains the two trials that timed out, two that produced solutions in just under three hours, and another which produced a solution in under 30 minutes. The variable behaviour within Primitive Set 13 can easily be considered an outlier, or an artefact, of that primitive set. However, the behaviour presents itself across Primitive Sets 11 to 16 and is witnessed more clearly in the results of *Tank* (Figure 4-13b) and *Tank-P4* (Figure 4-14b). *House-W* also exhibited variations of up to 30 minutes within the same primitive set (Primitive Sets 15 and 18), despite having a considerably smaller covering set compared to the *larger planning problems* (Figure 4-15).

These results suggest that as the covering set size increased, the disparity between results increased. Despite most of *Tank-P4* trials timing out, there were solutions that solved well below the time threshold. These findings are important as they highlight that solutions existed

well below what the average was indicating.

If there was to be any significant variations between the trials of a primitive set, it would be expected to be manifested in the overall tour lengths. As each of the 20 primitive sets generated the same configuration set across the five trials, the only variation that would be expected was a difference in tour length due to the random nature of the RRT solutions. As stipulated in Section 4.2.5, if a path cannot be solved via a direct connection, an RRT is invoked to resolve the path query. Due to the random expansion of an RRT, the likelihood of generating the exact path between the same two configurations is unlikely between different trials. Furthermore, as RRT paths are not solved optimally and have the potential to invalidate the triangular equality, it is understandable that the same tour order, and therefore tour length, may not be the same across all five trials.

The results in Tables 4-1 and 4-3 indicate there were minimal variation in tour lengths across all primitive sets and trials for both the *small* and *large planning problems*. The ratio of direct paths to RRT paths taken in the final tour solution can be seen in Table 4-4. Despite the complexity of *Tank* and *Tank-P4*, less than 1% of the final tours consist of RRT paths. Configuration sets in these planning problems are compact due to the visibility constraints that only allow the robot to observe a small portion of the overall environment. Due to the compactness, direct paths are likely to be taken between neighbouring configurations Consequently, only a small number of RRTs were required to be used in the final tours. However, given that no RRTs were used to solve *2x2m* and *6x6m* and the variability is still present, this eliminates the *RRT-Connect algorithm* as a potential cause behind the internal variation between trials of the primitive set. Therefore, despite the LPP exhibiting variable solution times, the variability does not impact the final solution.

The interesting finding is how little variability there is between the tour lengths for *Tank-P4* despite having the majority of the trials unable to complete within the allocated time. If the LPP required more time to solve, a larger disparity between tour lengths would be expected; the tour lengths for *Tank-P4* suggest this is not the case. An example is seen when examining the tour lengths of Primitive Set 4 of *Tank-P4* (Figure 4-14b). The only trial to complete within 1.5 hours, has a tour length of 174.19m. The remaining four trials averaged 174.26m with a standard deviation of 0.11. Therefore, 4.5 hours separates 0.07m for the same configuration set and this variation in planning could be caused by a difference in RRT paths selected in the final tour. It appears that a significant portion of the planning for these trials could have been solved

Table 4-4: Ratio of Direct and RRT paths used in the final tour.

Model	*Average number of paths in final solution*	*Final Tour Path Breakdown*			
		Direct		*RRT*	
		# Paths	*%*	*# Paths*	*%*
2x2m	*152.00*	*152.00*	*100.00*	*0.00*	*0.00*
6x6m	*1,011.05*	*1,011.05*	*100.00*	*0.00*	*0.00*
House	*321.80*	*320.20*	*99.50*	*1.60*	*0.50*
House-W	*325.35*	*322.62*	*99.16*	*2.73*	*0.84*
Tank	*1,301.44*	*1,295.00*	*99.51*	*6.44*	*0.49*
Tank-P4	*1,649.00*	*1,639.35*	*99.41*	*9.65*	*0.59*

within the first 1.5 hours as there is no significant difference in tour lengths that suggest the LPP is not converging to a better solution.

Table 4-5 shows the top and bottom three primitive sets exhibiting the smallest and largest difference in planning times between trials. These results indicate, especially for the *larger planning problems*, that the trials which take longer to plan are not guaranteed to be better or shorter solutions. The differences between the tour lengths, at either extremity, are so small that the minor variation between tour lengths can be due to the difference in the RRT paths solved between the tightly clustered covering sets. As mentioned previously, this can cause a slight tour reordering. These results indicate that between configuration sets and between trials, tour lengths are relatively consistent across each of the trials. Consequently, the variability in planning times must be caused by some other influence other than environmental factors.

Finally, the reason for using controlled planning scenarios was to evaluate specific aspects of the *offline coverage planner* against different environmental features. *2x2m* and *6x6m* were designed to understand how solving the CSP changes the size of the environment whilst limiting the influence of the LPP, as no obstacles are present. *House* and *House-W* were designed to limit the influence of the CSP to witness how the LPP responded to a small change in geometry that impacted potential path routes that were available for *House*. Table 4-6 lists a summary of all the expectations placed on the planning models and resultant outcomes from this experiment.

Of all the expectations, it was surprising to observe that the expectation placed on the LPP to terminate *2x2m* and *6x6m* after two planning iterations was the only exception not to be met. As seen in Table 4-7, neither planning problem terminated after two planning iterations and on average, evaluated more paths than expected. If the LPP was to terminate after two iterations,

Table 4-5: The difference between minimum and maximum within each set shows no correlation between longer planning times and shorter tour lengths.

Model	*Time Diff*	*P Set*	*Configs*	*Min Trial Time (s)*	*Max Trial Time (s)*	*Rel. Time Diff.*	*Min Trial Tour Length (m)*	*Max Trial Tour Length (m)*	*Rel. Tour Length Diff. (m)*
2x2m		*18*	*152*	*2.65*	*2.70*	*1.02*	*21.82*	*21.74*	*0.996*
	Min	*16*	*148*	*2.73*	*2.78*	*1.02*	*21.82*	*21.36*	*0.979*
		20	*149*	*2.56*	*2.62*	*1.02*	*21.82*	*21.74*	*0.996*
		7	*147*	*2.70*	*3.52*	*1.30*	*21.74*	*21.55*	*0.991*
	Max	*2*	*155*	*2.73*	*4.34*	*1.59*	*21.55*	*21.82*	*1.013*
		6	*152*	*2.74*	*4.38*	*1.60*	*21.82*	*21.55*	*0.988*
6x6m		*18*	*998*	*38.37*	*277.21*	*7.23*	*159.82*	*159.74*	*1.000*
	Min	*20*	*1002*	*28.87*	*277.16*	*9.60*	*160.65*	*160.98*	*1.002*
		9	*1019*	*29.21*	*300.43*	*10.29*	*159.74*	*160.65*	*1.006*
		12	*995*	*1,348.92*	*13,758.40*	*10.20*	*159.82*	*160.98*	*1.007*
	Max	*10*	*1018*	*464.99*	*16,740.60*	*36.00*	*159.74*	*160.98*	*1.008*
		13	*1017*	*1,232.18*	*21,622.80*	*17.55*	*160.98*	*159.82*	*0.993*
House		*13*	*328*	*14.72*	*15.20*	*1.03*	*33.39*	*34.56*	*1.035*
	Min	*10*	*321*	*4.70*	*5.69*	*1.21*	*33.75*	*33.60*	*0.996*
		5	*321*	*8.75*	*10.02*	*1.14*	*33.60*	*33.39*	*0.994*
		9	*326*	*15.89*	*33.15*	*2.09*	*33.75*	*34.56*	*1.024*
	Max	*4*	*319*	*5.32*	*24.50*	*4.60*	*34.56*	*34.77*	*1.006*
		1	*332*	*14.56*	*40.01*	*2.75*	*33.39*	*34.77*	*1.041*
House-W		*20*	*323*	*8,795.65*	*8,987.28*	*1.02*	*36.06*	*35.71*	*0.990*
	Min	*11*	*324*	*6,161.10*	*6,476.45*	*1.05*	*35.99*	*35.71*	*0.992*
		7	*324*	*6,987.30*	*7,325.64*	*1.05*	*35.33*	*35.99*	*1.019*
		6	*330*	*7,741.78*	*9,164.59*	*1.18*	*35.33*	*35.99*	*1.019*
	Max	*18*	*334*	*8,165.09*	*9,972.57*	*1.22*	*36.23*	*35.99*	*0.994*
		15	*330*	*9,081.56*	*11,056.80*	*1.22*	*36.06*	*35.71*	*0.990*
Tank		*8*	*1308*	*785.96*	*937.73*	*1.19*	*148.65*	*149.35*	*1.005*
	Min	*15*	*1311*	*425.60*	*1,328.39*	*3.12*	*148.65*	*147.08*	*0.989*
		19	*1300*	*591.75*	*2,558.03*	*4.32*	*148.65*	*148.63*	*1.000*
		3	*1311*	*2,078.32*	*21,634.10*	*10.41*	*148.63*	*147.01*	*0.989*
	Max	*11*	*1304*	*1,416.05*	*21,634.90*	*15.28*	*149.35*	*147.01*	*0.984*
		9	*1306*	*903.01*	*21,633.30*	*23.96*	*147.08*	*147.01*	*1.000*
Tank-P4		*3*	*1649*	*21,668.00*	*21,668.60*	*1.00*	*174.08*	*175.18*	*1.006*
	Min	*1*	*1665*	*21,666.40*	*21,667.10*	*1.00*	*175.39*	*174.08*	*0.993*
		10	*1653*	*21,666.00*	*21,666.70*	*1.00*	*175.18*	*176.66*	*1.008*
		6	*1661*	*15,838.40*	*21,667.50*	*1.37*	*174.08*	*178.03*	*1.023*
	Max	*5*	*1651*	*14,603.70*	*21,667.30*	*1.48*	*174.08*	*178.03*	*1.023*
		4	*1651*	*5,303.87*	*21,665.80*	*4.08*	*176.66*	*175.18*	*0.992*

Diff – Difference *Configs – Configurations* *Max – Maximum*
P Set – Primitive Set *Min – Minimum* *Rel - Relative*

Table 4-6: List of expected outcomes for each model set and the respective result from the experiment.

Model Set	Model Set Expectations	Expectation Met
2x2m and *6x6m*	Solving the CSP will take longer for *6x6m* than *2x2m*.	Yes
	The LPP will only take two iterations to solve *2x2m* and *6x6m* as the *Euclidean assumption* placed on the connectivity will always be correct.	No
	No RRTs are required to be validated to solve the MPP.	Yes
House and *House-W*	Solving with CSP should be approximately the same despite the inclusion of the change in *House-W*.	Yes
	The LPP will have to evaluate paths for *House-W* than *House* to compensate for the change.	Yes
All models	All complex models of each model sets will take longer to solve than the base models.	Yes

Table 4-7: Number of planning iterations and paths evaluated to solve each planning problem.

Model	*Planning Iterations*		*Path Evaluations*	
	$\bar{x}$	*SD*	$\bar{x}$	*SD*
2x2m	*4.87*	*4.47*	*157.73*	*8.92*
6x6m	*5,227.24*	*8,279.35*	*1,633.13*	*176.05*
House	*50.06*	*33.33*	*563.18*	*72.84*
House-W	*11,440.15*	*1,973.46*	*13,438.45*	*2,276.20*
Tank	*5,359.57*	*5,961.02*	*4,601.53*	*824.84*
*Tank-P4**	*16,928.72*	*2,151.91*	*5,814.68*	*522.23*

it would be expected that the TSP solver would produce identical results.

While all other expectations were true, a closer examination of the LPP planning data highlights some interesting findings. While *House-W* was expected to take longer than *House*, *House-W* performed over 12,500 more path evaluations to rectify the one small change that was introduced. This equated to as many as 11,000 more planning iterations and provided a reason why *House-W* took so long to solve and highlights how sensitive the LPP can be to small variations in environmental geometry. Furthermore, the average difference of four hours between planning times *Tank* and *Tank-P* can be due to the 1,200 additional paths *Tank-P* evaluated and the subsequent 10,000 additional planning iterations. Keeping in mind that the

majority of the trials for *Tank-P* did not finish in the allocated time, it is interesting that *House-W* evaluated more paths considering it was a smaller covering set and planning environment.

4.4.2 Discussion

The benchmark experiment has provided valuable insight into how the *offline sampling-based coverage planner* operates across the target and various constructed environments. The objectives of this experiment were to;

Objective 1) Benchmark the performance of the *offline sampling-based coverage planner* within a representative submarine tank environment.

Objective 2) Determine to what extent the LPP influences the planning times within the target environment.

Objective 3) Discover any other limitations of the coverage planner that may impact online performance.

The findings of this experiment have addressed all three of these objectives. This experiment has provided a benchmark of how the recreated implementation of the *offline sampling-based coverage planner* has performed in the target environment (*Objective 1*). The main findings of the benchmark showed:

1) The covering set sizes of the *larger planning problems*, in particular *Tank and Tank-P*, are well into the thousands. The covering sets generated in this experiment were significantly greater in size than presented originally in Englot (2012) and Englot and Hover (2013), highlighting the *offline sampling-based coverage planner* is capable of working well beyond the original implementation.

2) Significant planning times were observed for these environments. Planning times for *Tank* exceeded two-hours while 90% of the trials for *Tank-P* terminated due to the six-hour limit. These planning times are orders of magnitude greater than the largest planning problems presented in Englot (2012) and Englot and Hover (2013).

This experiment also demonstrated the extent to which the LPP dominated planning times (*Objective 2*). Planning times for *House*, *House-W*, *6x6m*, *Tank* and *Tank-P* all exhibited over 85% of their computational time solving the MPP. However, a further investigation into planning times highlighted a few concerns surrounding the behaviour of the LPP that will impact the performance of the LPP in an online situation (*Objective 3*). The statistical analysis

found that the LPP exhibited the following behaviours;

1) A significant variability between trials of the same covering set and that the variability is more apparent in larger covering sets,
2) small geometrical changes can have a large impact on planning times, and
3) the LPP did not terminate as expected, as shown the *2x2m* and *6x6m* examples.

These results suggest that the LPP is non-deterministic which is an undesirable trait for online planning. Minor variations in planning times could be explained by the differences in set size and path variation between trials, but the results illustrated in Figures 4-12 to 4-15 show there was no correlation of planning time variability to set size or tour length.

Despite this variability, it did not have an impact on final tour lengths. An analysis of the trials across all planning problems revealed that no correlation was present between longer planning times thus equating to better tour solutions. This indicates that that planning times can be reduced without compromising tour length. The minor variability exhibited in the tour length can be attributed to randomness of the RRT paths.

Other findings of this experiment included:

1) Solving the CSP made a small contribution to the overall planning times. The simplified constraints made sampling cheaper than what it would be for a high-DOF robot.
2) All planning problems achieved 100% coverage.
3) The sampling procedures were not found to exhibit any variability in solution times that would be concerning for online implementation.

The importance of these findings isolated the problems of the *offline sampling-based coverage planner* to be directly related to the LPP. The results provided by the CSP suggest that the CSP will reliably work in an online situation.

The issues to take away from this experiment is that the representative tank environments used were designed to expose the *offline coverage planner* to a smaller tank variation containing only some of the features expected within these environments. Actual submarine tanks are expected to be larger and will contain more features. Therefore, when implemented under real-world conditions, the *offline coverage planner* will naturally produce larger configuration sets than what were produced in this experiment. Given that the majority of *Tank-4* solutions terminated due to the six-hour time limit suggests that this implementation of the *offline*

sampling-based coverage planner is not currently suitable for online implementation. However, it was found that the reason for these excessive computational times was due to the LPP.

For an online implementation the above behaviours exhibited by the LPP create major concerns that need to be resolved for the LPP to be successful online. These findings are surprising as neither Saha and Latombe (2003), Englot (2012) or Englot and Hover (2012a, 2012b and 2017) report any variability in their results. The difference between these results and the results published previously, is that the planning environments used in this experiment created significantly larger MPPs, up to eight times larger, than those published previously.

While it was expected that all complex models of each pair would take longer than the base models (Section 4.3; Table 4-6), both the planning times and the planning data from the LPP suggests that the issue is intrinsic to the algorithms that form the LPP, given that no significant variability was evident in the tour lengths. The *2x2m* and *6x6m* planning problems suggest that this may be the case given that neither terminated within two planning iterations as expected. Furthermore, the results between *House* and *House-W*, demonstrated how vulnerable the LPP can be to a minor change in the environment but *Tank-P*, which on many occasions was terminated due to time, evaluated fewer paths on average. Finally, the results of *Tank* and *Tank-P* demonstrated the LPP's inability to solve the target application consistently within a six-hour time limit. Some trials for *Tank-P4* where able to solve the planning problem without timing out, indicating that there are underlying intrinsic issues causing this variability that are more than just the influence imposed by the environment. All these issues raise questions over the LPP applicability in an online setting.

Given these outcomes, the current LPP implementation is not suitable for any *online coverage planner* regardless of which *online replanning strategy* is implemented. Confidence in consistent and reliable results is important for online planning. If the LPP is to be used online, the variability needs to be resolved. In the next chapter, the planning data from the LPP is investigated in further detail to determine the cause behind the significant variability that was present across all planning problems and why *2x2m* and *6x6m* did not terminate as expected.

4.5 Chapter Summary

To summarise this chapter, the *offline sampling-based coverage planner* was recreated and benchmarked to determine how the coverage planner performed in a representative submarine tank environment. The *offline coverage planner* was recreated to perform the experiments in

this thesis, and a series of amendments were implemented to assist the *offline sampling-based coverage planner* to work effectively in complex planning environments. The amendments included the introduction of a *primitive rejection count* to limit the number of times a primitive is sampled to avoid oversampling in complex spaces.

To investigate the intrinsic functionality of the CSP and MPP methods, the simplified motion planning and visibility constraints discussed in *Chapter 3* were used to reduce the impact of the extrinsic influences of a multi-legged, high-DOF platform would have on the planning procedures. A series of controlled environments were tested alongside two representative tank environments to benchmark the performance of the *offline sampling-based coverage planner*.

The main findings of the benchmark experiment demonstrated that;

1) planning within the representative tank environments is expensive, requiring thousands of configurations to solve the CSP,
2) the LPP dominated all but one planning problem,
3) the LPP exhibited an unexpected variability across all trials that inhibited the same covering set to be solved in a similar time,
4) the larger the covering set grew in size, the more significant the variability was between solution times.
5) variability was not witnessed in CSP times or the final tour lengths, and
6) the LPP did not terminate as expected.

Overall, the experiment achieved its intended objectives by providing information about the coverage planner that highlights that the current implementation of the *offline sampling-based coverage planner*, more specifically the LPP, was not suitable for online implementation. Given the importance of delivering stable and consistent solutions in an online situation, the next chapter investigates the planning data of the LPP to determine the cause of the significant variability that is present amongst all trials.

Chapter 5
Minimising the Variability of the LPP for Stable Online Solutions

5.1 Introduction

The results of the benchmark experiment of *Chapter 4* highlighted that the excessive planning times and significant variability associated with the *lazy point-to-point planner* (LPP) need to be resolved. The results strongly indicated that the environment had a considerable influence on the planning times. However, the variation between identical configuration sets indicated the issue was intrinsic to the algorithmic processes inside the LPP. In this chapter, the planning data acquired from the LPP during the benchmark experiment was analysed to determine the cause of the significant variation present in all planning problems.

5.2 Removing the $O(n^2)$ Complexity as a Factor of Computational Variability

The main concern with using the LPP was the complexity of the algorithm which at worst is $O(n^2)$. Considering the large configuration set sizes generated over the *larger planning problems* (*6x6m, Tank and Tank-P*), it could have been likely that the excessive solution times may being a consequence of the LPP searching through more of the solution space to find an answer.

Comparing the average number of path evaluations against the average number of total paths in the planning problem showed that no planning problem came close to evaluating to the limit of $O(n^2)$ (Table 5-1), i.e. all possible path combinations. Besides *House-W*, which evaluated 25.5% of the total paths, the remaining planning problems evaluated no more than 1.5% of their respective solution spaces. Even *Tank-P4*, which prematurely terminated, only evaluated

Table 5-1: *Lazy point-to-point planner* planning data for all planning problems.

Model	*Average of total number of paths in problem (TP)* $^{\Phi}$	*Planning iterations (IT)*					*Path evaluations (PE)*					*PE / TP (%)*
		$\bar{x}$	*SD*	*Median*	*IQR*	*CV*	$\bar{x}$	*SD*	*Median*	*IQR*	*CV*	
2x2m	*11,480.35*	*4.87*	*4.47*	*4.00*	*4.00*	*91.81*	*157.73*	*8.92*	*155.00*	*13.00*	*5.65*	*1.38*
6x6m	*510,637.75*	*5,227.24*	*8,279.35*	*2,001.50*	*5,001.50*	*158.39*	*1,633.13*	*176.05*	*1,646.00*	*267.50*	*10.78*	*0.32*
House	*51,631.25*	*50.06*	*33.33*	*45.00*	*43.50*	*66.57*	*563.18*	*72.84*	*574.00*	*109.50*	*12.93*	*1.09*
House-W	*52,775.50*	*11,440.15*	*1,973.46*	*11,484.00*	*3,343.00*	*17.25*	*13,438.45*	*2,276.20*	*13,578.50*	*4,216.00*	*16.94*	*25.48*
Tank	*846,664.55*	*5,359.57*	*5,961.02*	*2,413.00*	*7,377.00*	*111.22*	*4,601.53*	*824.84*	*4,492.50*	*1,205.00*	*17.93*	*0.54*
*Tank-P4**	*1,368,211.30*	*16,928.72*	*2,151.91*	*17,480.00*	*16.00*	*12.71*	*5,814.68*	*522.23*	*5,787.50*	*584.00*	*8.98*	*0.42*

** Average taken across 50 trials*
$^{\Phi}$ Number of paths in problem is equal to (n (n +1) / 2) - n where n is the number of configurations. A lower diagonal matrix is all that is needed for symmetrical Travelling Salesman Problems
^ Average path evaluations per iteration taken after the first iteration. The first iteration will always evaluate the maximum number of paths in a planning problem
IQR - Interquartile Range
CV - Coefficient of Variation

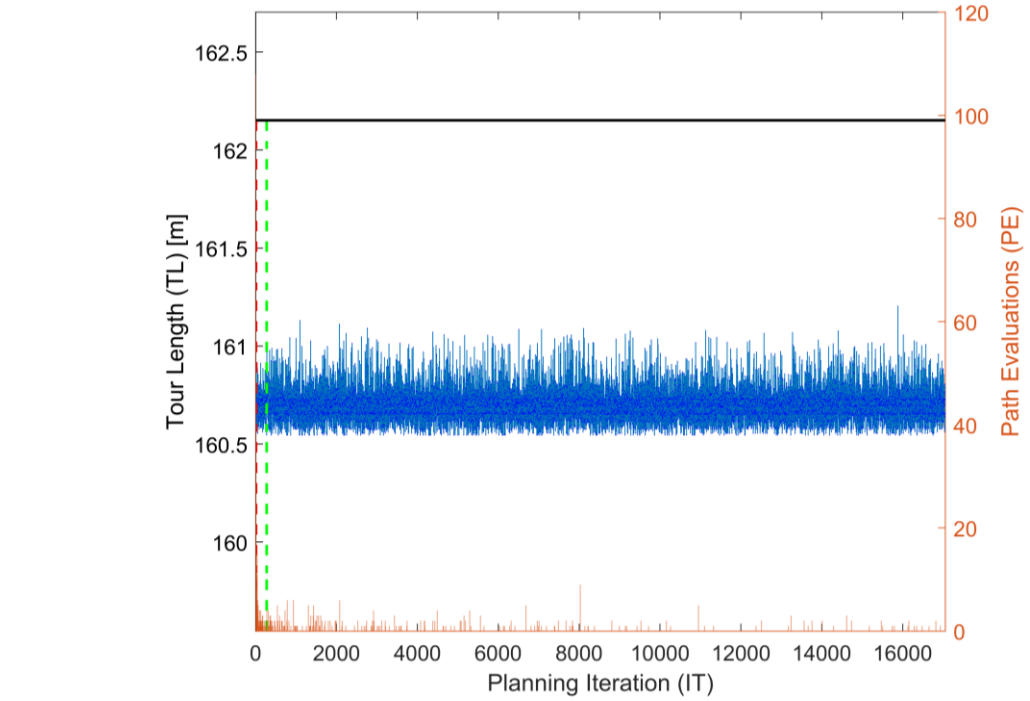
a small proportion of the total solution space. This finding ruled out the possibility, that the times to solve the *multi-goal planning problem* (MPP) were not a result of solving the solution space to the worst-case complexity, therefore confirming previously made observations about the LPP.

5.3 Variable Planning Iterations

The statistical analysis shown in Table 5-1 indicated that the planning iterations were highly variable. Given that the number of paths evaluated was dependent upon the solutions provided by the *Travelling Salesman Problem* (TSP) solver, suggests that the variability may be due to limitations in the TSP solver.

These limitations are highlighted in the *2x2m* and *6x6m* examples. In the Section 4.3.1, it was hypothesised that both *2x2m* and *6x6m* would always terminate after two iterations, regardless of the covering set size, as the *Euclidean assumption* would never have to be updated. As no RRTs would have been required to resolve any path queries, the resultant adjacency matrix remained the same, so it would be expected that the TSP solver would provide identical solutions on consecutive iterations. However, as recorded in Table 4-7 and in Table 5-1, both planning problems executed a larger number of planning iterations and evaluated a larger number of paths than configurations. For *6x6m*, this led to planning times ranging from 45 minutes up to six hours, executing an additional 5,225 planning iterations than expected. Considering no paths were evaluated by an RRT, the variability observed in the number of planning iterations for *2x2m* and *6x6m* suggested that the TSP solver had difficulty stabilising a solution and therefore terminating as expected.

Evidence of the LPP's inability to stabilise was present in a single trial of the *6x6m*, shown in Figure 5-1. In this trial, it was expected that the *First-Zero Path Evaluation Iteration* (FZEI) would occur at Iteration 2, however, it occurred at Iteration 9. This result showed that the TSP solver did not produce identical solutions in iterations for the same adjacency matrix. Due to this unrepeatability the subsequent solutions did not trigger the *equality termination condition* after the second iteration of the LPP (Algorithm 3-2, Line 4). The trial presented in Figure 5-1 highlights that the TSP solver continued to analyse alternative paths that resulted in new tours being formed that were different to previous iterations. Consequently, it caused the LPP to fluctuate between several thousand solutions until the TSP solver eventually found two identical tours to terminate the LPP at Iteration 17,056**.**



Summary	
Final TL (IT)	*160.58m (17,056)*
BTL (IT)	*160.55m (273)*
FZEL (IT)	*160.66m (9)*
# TL within 1% of the BTL (% of Total ITs)	*17,056 (100%)*
Repeated ITs	*28*
Total PEs	*1,806*
# PE after the FZEL (% of Total PEs)	*531 (29.40%)*

Figure 5-1: Lazy point-to-point planning data for a trial of 6x6m.

Further analysis of the *6x6m* trial in Figure 5-1 uncovered additional behaviours that were contrary to expectations. The following behaviours were observed;

1) *The Final Tour Length* (FTL) is not the *Best Tour Length* (BTL) found (160.58 vs 160.55m).
2) The LPP did not recognise the BTL and failed to terminate in the subsequent iteration.
3) The *FZE Tour Length* (FZEL), which was expected to terminate the LPP with the BTL, was not equal to the FTL (160.66 mm vs 160.58).
4) The FZEL was found earlier than the BTL.
5) The TSP solver produces previously solved solutions but not necessarily in consecutive iterations.
6) As result of not terminating when expected, a significant number of iterations occur that do not evaluate new paths, denoted as a *Zero Path Evaluation Iterations* (ZEIs), but produce different tours, i.e. different tour order.

The analysis of all the trials indicated that these behaviours were present over all planning problems (Tables 5-2, 5-3 and 5-4) and had a greater impact on the *larger planning problems*. Figures 5-2 and 5-3 illustrate the listed behaviours from a single trial of *Tank* and *Tank-P4*, respectively. The consequences of failing to terminate, resulted in the additional time and computational effort spent searching for alternative solutions that did not always produce a better result. As path evaluations under simplified constraints were computationally efficient, reducing computational effort per iteration, the majority of the LPP computation time was due to excessive number of iterations generating TSP solutions.

A surprising finding was that the FTL was not equal to the BTL. These results presented in Table 5-2 show that there was no significant difference between the average FTLs and BTLs for all planning problems as all FTLs are within 1% of the BTL. However, the final two columns of Table 5-2 indicate the fundamental problem; *the TSP fails to detect the BTL as the best solution in consecutive iterations therefore forcing the LPP to continue searching for new solutions*. Despite finding the BTL, the LPP continued to iterate through more solutions for no further improvements upon the current best. Several planning problems reported that over 50% of their total planning iterations occurred after the BTL had been found, and during that time, very few paths were evaluated. Consequently, this contributed to the significant number of ZEIs present in the results.

An interesting and important finding which was central to determining the cause of the variability, was the number of planning iterations that returned a ZEI. For the *larger planning problems*, over 50%-90% of planning iterations evaluated no paths (Table 5-3). However, as discussed previously, *2x2m* and *6x6m* should have terminated upon the FZEI.

When a planning iteration reports a ZEI there are two possibilities;

Possibility 1) the TSP solver has produced a tour that is identical to a previous tour such that all the same paths, in order, have been evaluated in a previous iteration, and

Possibility 2) the TSP solver has produced a tour that is an amalgamation of evaluated paths from previous tours that collectively form a new tour.

For *House*, *House-W*, *Tank* and *Tank-P* it is not surprising that they exhibited both *Possibility 1* and *2*. Given that paths are only evaluated once and the covering sets are quite dense, slight permutations to the tour, as the LPP begins to stabilise, can be expected to

Table 5-2: A comparison between best tour length and final tour length highlight there is no significant difference between the two solutions.

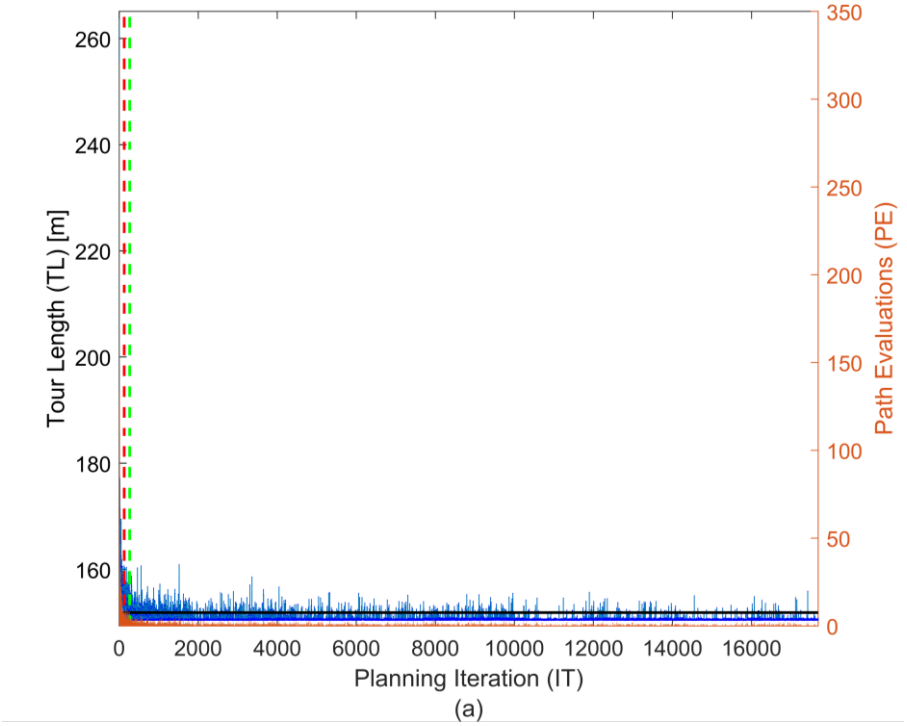
Model	*Average FTL (m)**	*Average BTL (m)**	*Difference between the BTL and FTL ($x10^{-3}$ %)^*		*Number of FTLs within 1% of the BTLs as % of total trials (%)^*	*Average number of IT after the BTL as a % of total IT (%)^*	*Average number of PE after the BTL as a % of total PE (%)^*
			$\bar{x}$	*SD*			
2x2m	*21.47*	*21.47*	*0.85*	*4.29*	*100.00*	*59.50*	*2.14*
6x6m	*159.46*	*159.40*	*37.74*	*32.54*	*100.00*	*69.12*	*8.14*
House	*33.94*	*33.94*	*1.44*	*4.34*	*100.00*	*24.88*	*1.33*
House-W	*35.72*	*35.70*	*60.18*	*134.35*	*100.00*	*6.57*	*5.26*
Tank	*148.99*	*148.93*	*44.74*	*51.48*	*100.00*	*52.80*	*9.55*
Tank-P4	*175.65*	*175.52*	*72.69*	*42.68*	*100.00*	*46.14*	*5.99*

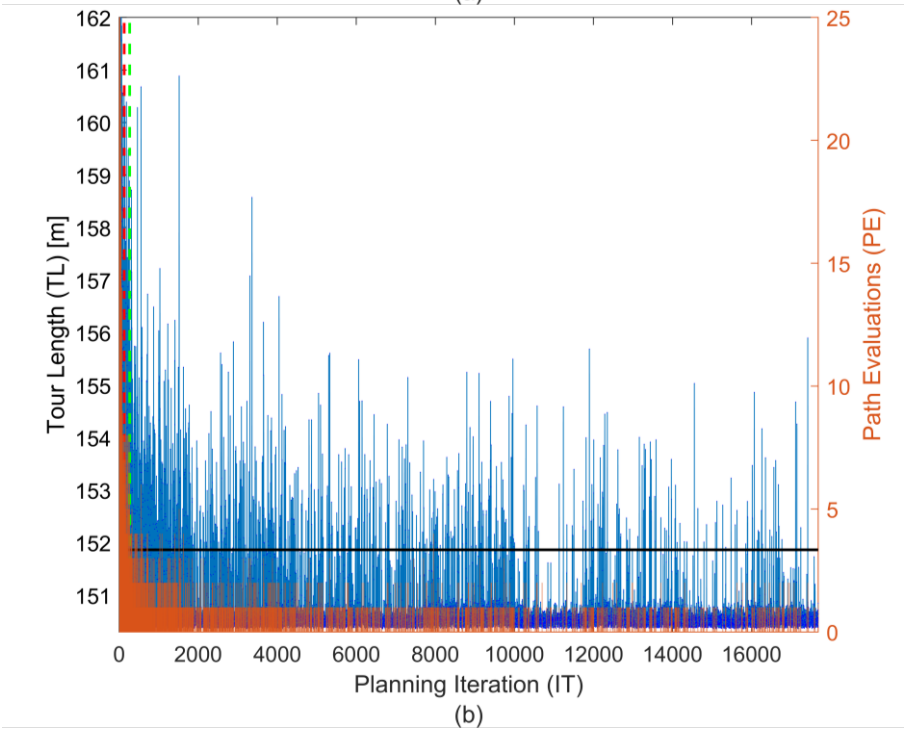
** Average taken over all trials*
^ Average of the relative difference between values within each trial
FTL - Final Tour Length
BTL - Best Tour Length
IT - Iterations
PE - Path Evaluations

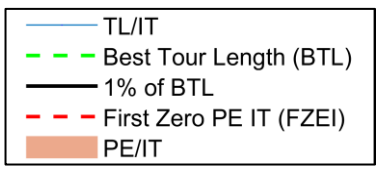
Table 5-3: The presence of iterations that evaluated no new paths or repeat previous solutions is prevalent throughout every planning problem.

Model	*Average IT to solve problem**	*Average number of ZEIs as a % of total IT (%)^*	*Average number of IT after the FZEI as a % of total IT (%)^*	*Average number of PE after the FZEI as a % of total PE (%)^*	*Number of ZETL within 1% of the BTLs as a % of total ZEIs (%)^*		*Number of ZETL within 1% of the FTLs as a % of total ZEIs (%)^*		*Average FZEL (m)**	*Difference between the FZEL and FTL (%)^*	
					$\bar{x}$	*SD*	$\bar{x}$	*SD*		$\bar{x}$	*SD*
2x2m	*5.25*	*28.74*	*30.33*	*1.20*	*100.00*	*0.00*	*100.00*	*0.00*	*21.47*	*0.01*	*0.02*
6x6m	*5,253.74*	*83.15*	*95.64*	*20.22*	*100.00*	*0.00*	*100.00*	*0.00*	*159.48*	*0.01*	*0.04*
House	*49.41*	*21.65*	*28.48*	*1.81*	*100.00*	*0.00*	*100.00*	*0.00*	*33.94*	*-0.01*	*0.01*
House-W	*11,350.58*	*1.05*	*11.56*	*9.71*	*98.25*	*4.40*	*98.46*	*4.12*	*35.72*	*0.01*	*0.18*
Tank	*5,349.06*	*58.24*	*88.77*	*28.08*	*100.00*	*0.00*	*100.00*	*0.00*	*148.99*	*-0.01*	*0.06*
Tank-P4	*17,266.08*	*90.46*	*99.20*	*36.07*	*100.00*	*0.00*	*100.00*	*0.00*	*175.61*	*-0.02*	*0.05*

**Average taken over all trials*
^Average of the relative difference between each trial
ZEI - Zero-path evaluation iteration
FZEI - First ZEI
ZETL - Zero-path evaluation tour length
FZEL - First ZETL
FT - Final tour length
BTL - Best tour length
IT - Iterations
PE- Path evaluations

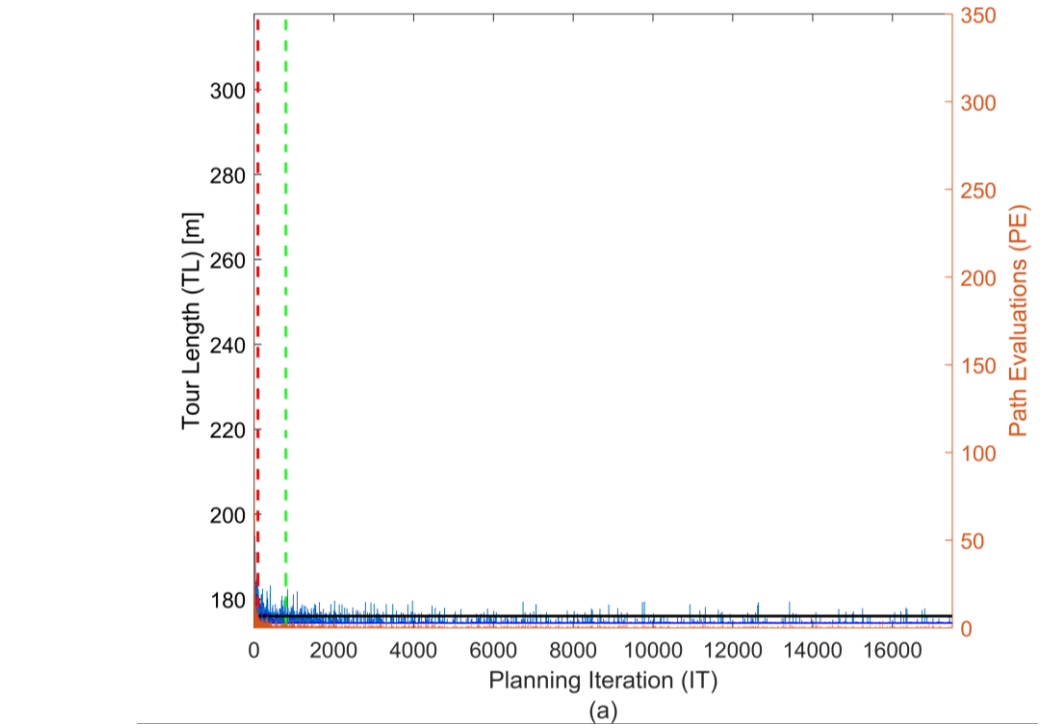


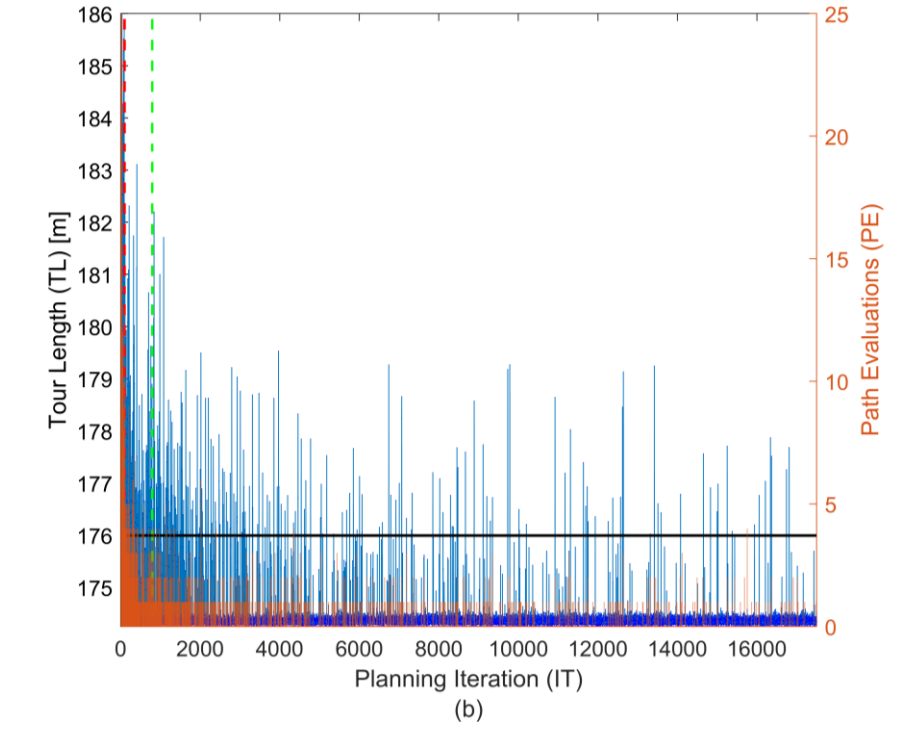


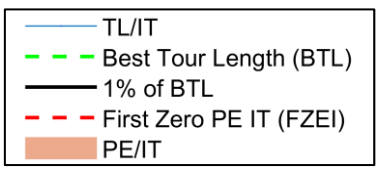


Summary	
Final TL (IT)	*150.65m (17,689)*
BTL (IT)	*150.36m (265)*
FZEL (IT)	*150.47m (126)*
# TL within 1% of the BTL (% of Total ITs)	*16,638 (94.06%)*
Repeated ITs	*565*
Total PEs	*5,739*
# PE after the FZEL (% of Total PEs)	*2,486 (43.32%)*

Figure 5-2: Lazy point-to-point planning data for a trial of *Tank*. (a) All planning data. (b) A closer examination of the planning data.







Summary	
Final TL (IT)	*174.34m (17502)*
BTL (IT)	*174.25m (789)*
FZEL (IT)	*174.29m (95)*
# TL within 1% of the BTL (% of Total ITs)	*17047 (97.4%)*
Repeated ITs	*11*
Total PEs	*4890*
# PE after the FZEL (% of Total PEs)	*1510 (30.88%)*

Figure 5-3: Lazy point-to-point planning data for a trial of *Tank-P4*. (a) All planning data. (b) A closer examination of the planning data.

Table 5-4: The average number of repeated and unique solutions for each planning problem.

Model	*Average IT to solve problem**	*Average number of repetitions as % of total IT (%)^*	*Average number of repeated solutions**	*Average number of unique solutions**
2x2m	*5.25*	*21.18*	*0.85*	*1.92*
6x6m	*5,253.74*	*16.64*	*492.35*	*4,058.76*
House	*49.41*	*13.83*	*2.91*	*40.25*
House-W	*11,350.58*	*0.65*	*16.69*	*11,361.37*
Tank	*5,349.06*	*10.93*	*387.47*	*4640.45*
Tank-P4	*17,266.08*	*1.38*	*159.14*	*16,704.74*

^Average of the relative difference between values within each trial *IT - Iterations*
**Average taken over all trials*

report a ZEI. The expectation was when a FZEI occurs, as FZIs do not change the adjacency matrix, the TSP solver should soon stabilise on a solution and terminate the LPP. However, in many trials, their termination did not occur soon after the FZEI and did not terminate with the BTL. Again, highlighting the non-deterministic behaviour that the LPP demonstrated in the benchmark experiment due to the inability of the TSP solver to stabilise on a solution.

It became clear that once the LPP encountered the FZEI, the LPP entered a *state of minimal improvement* until a time when two successive iterations were equivalent. It was during this period the TSP failed to effectively terminate the LPP resulting in several ZEIs occurring across all planning problems. Further analysis into each ZEI for every trial, over all planning problems, showed interesting properties of ZEIs. These properties are listed as follows:

1) The FZEI can occur significantly earlier than the BTL for all planning problems besides *2x2m* and *House* (Table 5-3, Column 4).
2) Tour lengths produced by ZEI (ZETL) are within 1% of the FTL and BTL (Table 5-3, Columns 6 and 7), thus highlighting that beyond the FZEI the planner is indeed in *a state of minimal improvement*.
3) ZETLs and FTLs were approximately equivalent, with FZTLs producing marginally better results, for *House*, *Tank* and *Tank-P* (Table 5-3, Column 9). This finding again highlighting that all the extra computation does not create a better outcome.
4) Once the LPP enters a *state of minimal improvement*, the LPP fluctuates between a set of reasonable solutions. In the case of the *smaller planning problems*, the set of possible paths are small enough that they will repeat. In the *larger planning problems*, they are large enough to generate a series of unique solutions that it is unlikely repeat often (Table 5-4). This behaviour was not expected and highlights the

extent to which the TSP solver is influenced by the size of the covering set.

Given this analysis, it is clear that the *state of minimal improvement* needs to be resolved otherwise *the LPP is not suitable for online implementation*. If the variability cannot be resolved, an alternative MPP planner may be required. The following section describes the cause of the behaviours observed by the TSP solver, to determine if replacing the LPP is a suitable course of action.

5.4 Reasons Why the TSP Solver Produced Variable Solutions

The TSP variability is due to the manner in which the TSP solver is applied in this context. As previously discussed in Section 4.2.4, the LPP implements a *Quick-Borůvka* (QB) and *Chained Lin-Kernighan* (CLK) combination (QB-CLK) to solve the TSP. Both the QB and the CLK are *approximation algorithms* that use heuristics to generate solutions for the TSP. (Helsgaun, 2000; Applegate et al., 2003). It is a characteristic of approximation algorithms that they cannot guarantee an optimal solution, however a good solution can be found close to the optimum (Helsgaun, 2000).

Generally, to find the optimal solution using *tour improvement heuristics*, such as the CLK, may require the CLK to be run several times with different randomised tours (Helsgaun, 2000). For planning problems that comprise up to 50 cities, optimal answers could be found in the first iteration. For larger planning problems consisting of over 100 cities, the probablity of finding the optimal tour decreases to 20-30% and additional trials that randomise the initial tour provides a better opportunity of finding the optimal answer (Helsgaun, 2000).

The CLK, improves the chances of finding optimal answers by using '*kicks'* to slighly perturb the tour instead of restarting with a random tour (Applegate et al., 2003), equivalent to adding noise to the data to avoid local minima as in the case of machine learning (Burton Jr and Mpitsos 1992; Treadgold and Gedeon, 1998). To solve larger TSP problems, the CLK is supplied with an initial TSP solution supplied by an *approximation tour construction algorithm*, such as the QB (Helsgaun, 2000). The combination of the QB-CLK enables the TSP for set sizes ranging from 10,000 to 100,000 to be solved to near optimality (Applegate et al., 2003).

The current implementation of the LPP does not factor into account that the TSP solutions

are sub-optimal. The current *equality termination condition* requires that solutions are consistantly generated by the TSP solver, especially when the adjacency matrix is unchanged. The CLK is only given one trial per iteration to find a solution and is therefore constrainted to find a solution within a specified time limit. This limits the possiblity that the QB-CLK will produce an optimal solution. As a result, the LPP essentially terminates at random as the chances of terminating due to the *equality termination condition* significantly decreases as the covering set size becomes larger.

As Englot (2012) did not solve covering sets to the size of the *larger planning problems*, these behaviours exhibited by the LPP may not have been detected or have not been an issue worthy of consideration. Even when Englot and Hover (2012b) substituted the *Christofides*-CLK TSP solver for the *Nearest Neighbours-CLK* implementation, another *approximation tour construction algorithm*, there were no published reports of this substitution causing the LPP to exhibit variable solutions that prevent the LPP from terminating. Smaller TSP problems inherently contain smaller path permutations making it possible to find the optimal solution in a given QB-CLK iteration. The *smaller planning problems* (*2x2m*, *House* and *House-W*) used in the benchmark experiment, did not exhibit significant variability that would have indicated planning times were being impacted by this issue. This behaviour could have easily gone unnoticed if not for the *2x2*m and *6x6m*, which demonstrated that the LPP did not terminate as expected, therefore pointing to the influence that the TSP solver had on *House* and *House-W*. Without this insight, it would have been assumed the difference between these two environments would have been due to environmental influences.

In summary, *the benchmark experiment has uncovered the source behind the LPP's variability to be the approximation tour construction and tour improvement heuristics that were used to solve the TSP*. The analysis of the LPP planning data has shown that while the *smaller planning problems* could trigger the *equality termination condition* due to the QB-CLK ability to find optimal solution for smaller size of the problems, the current termination condition is not effective for planning problems of larger sizes.

Despite the evidence suggesting that replacing the QB-CLK with a non-approximation solver is a suitable solution to remove the *state of minimal improvement*, it also showed that the QB-CLK functions suitability well to converge the LPP to a set of near-optimal solutions. The analysis showed that if the s*tate of minimal improvement* is removed, the planning

problems could have been solved more quickly. As the effectiveness of the *equality termination condition* is impacted by the QB-CLK, if the termination condition were to be relaxed so the LPP terminated upon entering the *state of minimal improvement*, it would ensure the effectiveness of the QB-CLK in both offline and online planning for large planning problems.

5.5 Relaxing the Equality Termination Condition to Compensate for Near-Optimal TSP Solutions Generated by Approximation TSP Solvers

To compensate for the variations, *additional termination conditions* are added to the LPP to relax the *equality termination condition* so the LPP can terminate if consecutive planning iterations continue to make no meaningful contribution towards finding a better solution. These *additional termination conditions* do not replace the *equality termination condition* but provide a way to track the status of the LPP and terminate with the best possible solution before the planner enters a *state of minimal improvement*.

The analysis of the lazy point-to-point planning data (Section 5.3) indicated that the LPP entered a *state of minimal improvement* immediately after the LPP encountered a FZEI, whereupon the LPP exhibits two distinct behaviours (Table 5-4):

1) the LPP oscillates between previously solved solutions in non-consecutive iterations.
2) the LPP generates a series on new unique solutions without evaluating any new paths.

In both cases, these behaviours produce several ZEIs that generally do not contribute a significant improvement. Based on the results presented in Tables 5-3 and 5-4, the likelihood of any significant improvement occurring after encountering the FZEI is low. The *additional termination conditions* were designed to track the status of the LPP and terminate when either one of these events occurred.

As discussed in Section 5.3, an FZEI can be a tour that is constructed from previously evaluated paths (*Possibility 2*). Therefore, as it may be a new solution, termination of the planner upon the FZEI, or any subsequent ZEI, does not guarantee the LPP has finished. To ensure the LPP is not terminated prematurely, additional time should be given, if only for a short period, to find a better tour length than the current BTL. Therefore, the *additional termination conditions* were only activated to track the status of the LPP after the FZEI is recorded. At this time, the *additional termination conditions* are only considered if;

1) the original *equality termination condition* has failed, and
2) the current iteration has reported a ZEI.

To enable the use of the *additional termination conditions*, the LPP was modified to record each of the tour lengths at each planning iteration so the convergence into the *state of minimal improvement* can be tracked. By maintaining a record of all solutions, the BTL that is found can always be provided as the final solution; an expected outcome of the original implementation that was found not to be true.

Figure 5-4 illustrates the following description of how the *additional termination conditions* were implemented inside the LPP. In the event that the TSP begins to repeat previous solutions in non-consecutive iterations, the *current tour length* (CTL) is first compared against all the previously recorded solutions to determine if it has been solved before. If the CTL is indeed a repetition of a previous tour, the CTL is compared to the tour length to *previous tour length* (PTL) and if the CTL is less than the PTL, the LPP is terminated and the BTL becomes the final solution.

When the LPP enters a *state of minimal improvement*, characterised by producing several non-repeating ZEIs, the solutions are checked to determine if they are within 1% of the BTL found so far. If successive iterations are within 1% of the BTL, the *additional termination conditions* are triggered, and the BTL is presented as the final solution.

To ensure the LPP does not terminate prematurely when tracking non-repeating ZEI solutions, a limit of three successive, non-repeating ZEIs is set. These ZEIs must be less than the previous iteration to be sufficient to assert that the TSP solver has stabilised to a solution (*1% limit counter*) and terminate. If at any stage, an iteration returns a tour length better than the current BTL, the *1% limit counter* is reset.

While the *additional termination conditions* are predominantly targeted at the *larger planning problems*, by relaxing the original *equality termination condition*, the *additional termination conditions* become applicable to any planning problem of any size and environment. The *additional termination conditions* do not require changes to the TSP solver, which still provides solutions efficiently, while compensating for the non-optimal solutions exhibited by algorithms that approximate the TSP.

The option of relaxing the constraints is considered a better solution than placing a shorter

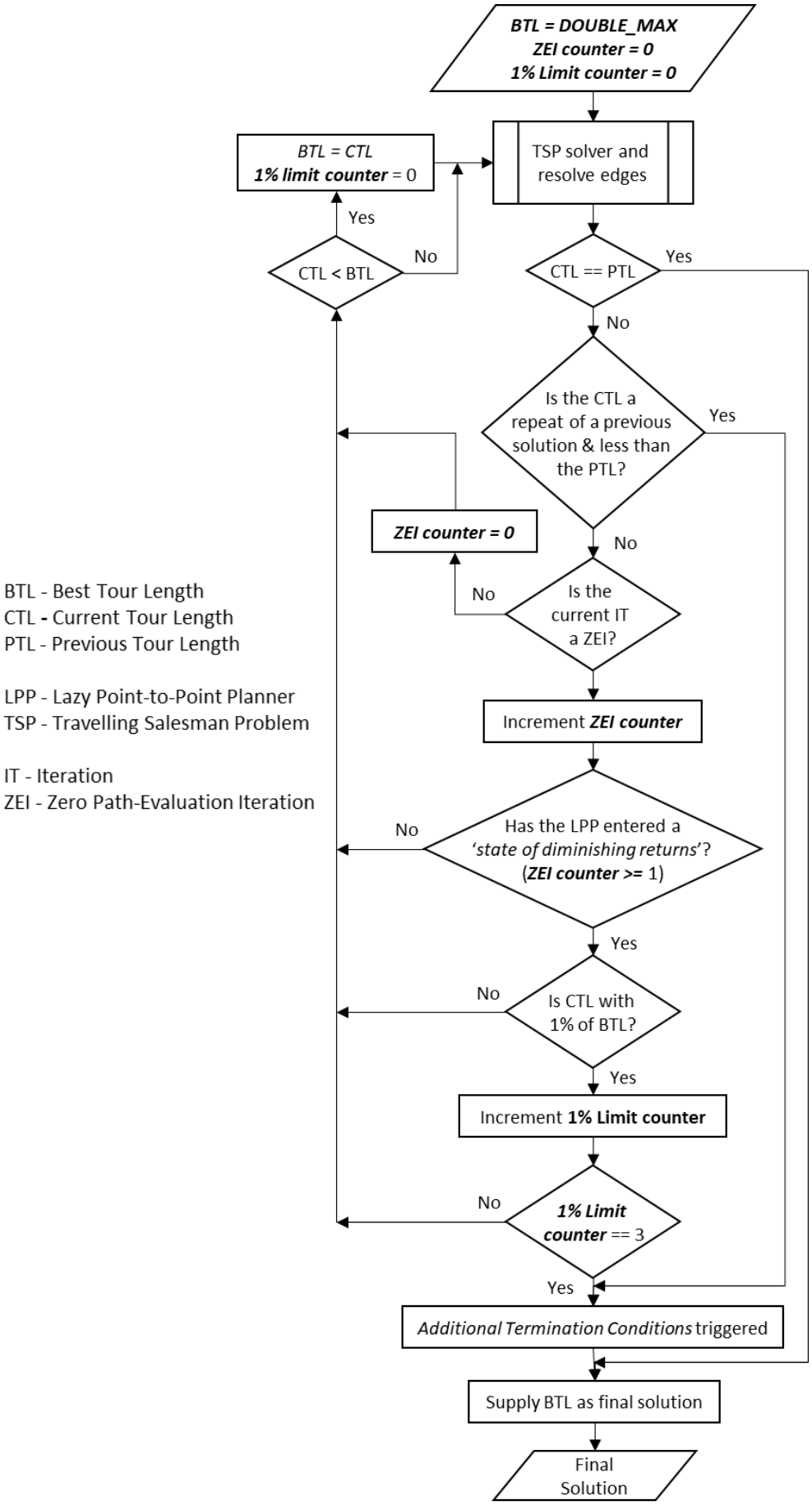


Figure 5-4: Flowchart of how the *additional termination conditions* are integrated into the LPP.

time-limit on the LPP. The benchmark experiment showed that the LPP responded differently to each planning environment, making it difficult to correctly infer a suitable time that ensured the planner found a feasible solution. By tracking the progress and behaviour of the LPP, the *additional termination conditions* provide an opportunity for the LPP to find the best solution before it is terminated.

5.6 Coverage Planning with the Additional Termination Conditions

To determine the effectiveness of the *additional termination conditions*, the benchmark experiment in Section 4.4 was performed again with the new conditions added to the LPP. To create a new benchmark of the *offline-sampling-based coverage planner*, the experiment was rerun over the same 20 random primitive sets to produce the same covering sets from the *redundancy-ten roadmaps*. Five trials were again applied to each primitive set. By running the same trials again, a direct comparison was made between the two experiments. For consistency the six-hour time limit was again enforced.

Pairwise comparisons were used to investigate whether a statistically significant difference (p) was present between the trials of the benchmark experiment and the trials using the new termination conditions for all planning attributes. For normally distributed planning attributes of the benchmark experiment, a *parametric Paired Samples T-Test* was conducted ($\alpha = 0.01$). For planning attributes of the benchmark experiment that were not normally distributed, a *non-parametric Sign Test* ($\alpha = 0.01$) was used. *Cohen's d* effect size (*d;* Cohen, 2013) was recorded for any attributes that demonstrated a statistically significant result. To allow an unbiased comparison between each experiment, all trials were conducted on the same 64-bit Intel i7 920 CPU, 8 core, 6GB RAM machine running Ubuntu 16.04 LTS as used in the benchmark experiment. All data analysis was performed using MATLAB 2018b.

5.6.1 Computational Observations and Results

The introduction of the *additional termination conditions* has aided in stabilising the LPP to produce consistent solution times across all planning problems without compromising tour quality. The statistical analysis of the overall, CSP and MPP planning times are presented in Table 5-5 and the analysis of the LPP planning attributes are presented in Table 5-6.

Of the 600 trials conducted in this experiment, 85% terminated due to the *additional termination conditions* with 15% being resolved using the *equality termination condition* (Table 5-7). No trials terminated due to the six-hour time limit. Even *Tank-P4*, which had

Table 5-5: Planning times for all planning scenarios using the *additional termination conditions.*

Model	*Trials*	*Overall Time (s)*				*CSP Time (s)*				*MPP Time (s)*			
		$\bar{x}$	*SD*	*median*	*IQR*	$\bar{x}$	*SD*	*median*	*IQR*	$\bar{x}$	*SD*	*median*	*IQR*
2x2m	*100*	*2.75*	*0.10*	*2.74*	*0.11*	*2.55*	*0.04*	*2.55*	*0.06*	*0.19*	*0.08*	*0.18*	*0.09*
6x6m	*100*	*38.44*	*5.60*	*39.11*	*6.63*	*21.94*	*0.74*	*21.94*	*0.90*	*16.41*	*5.67*	*16.72*	*6.95*
House	*100*	*11.62*	*5.45*	*10.76*	*9.00*	*2.60*	*0.03*	*2.60*	*0.03*	*9.00*	*5.46*	*8.13*	*9.02*
House-W	*100*	*7,044.92*	*1,444.96*	*7,161.66*	*2,549.36*	*2.65*	*0.03*	*2.65*	*0.04*	*7,042.25*	*1,444.95*	*7,158.98*	*2,549.39*
Tank	*100*	*429.56*	*123.79*	*419.50*	*179.78*	*32.52*	*0.39*	*32.52*	*0.57*	*396.98*	*123.71*	*386.62*	*180.24*
Tank-P4	*100*	*400.48*	*115.28*	*388.26*	*158.42*	*65.74*	*0.91*	*65.69*	*1.26*	*334.65*	*114.84*	*321.75*	*156.56*

CSP - Coverage Sampling Problem *MPP - Multi-goal Planning Problem* *IQR - Interquartile Range*

Table 5-6: *Lazy point-to-point planner* attributes for all planning scenarios using the *additional termination conditions.*

Model	*Tour Length (m)*				*Iterations*				*Path Evaluations*			
	$\bar{x}$	*SD*	*median*	*IQR*	$\bar{x}$	*SD*	*median*	*IQR*	$\bar{x}$	*SD*	*median*	*IQR*
2x2m	*21.47*	*0.27*	*21.49*	*0.53*	*2.90*	*1.20*	*3.00*	*1.00*	*156.40*	*8.16*	*154.00*	*6.00*
6x6m	*159.42*	*1.06*	*159.72*	*1.66*	*31.44*	*10.99*	*32.00*	*13.50*	*1,363.76*	*73.36*	*1,367.00*	*104.00*
House	*33.94*	*0.56*	*33.94*	*0.89*	*37.86*	*23.80*	*35.00*	*36.00*	*557.62*	*72.02*	*563.50*	*104.50*
House-W	*35.70*	*0.47*	*35.68*	*0.62*	*10,657.31*	*1,813.97*	*10,919.00*	*3,107.50*	*12,717.84*	*2,153.55*	*12,851.00*	*3,593.50*
Tank	*148.94*	*1.08*	*148.81*	*1.18*	*319.78*	*100.36*	*311.00*	*146.50*	*3,594.35*	*309.96*	*3,595.50*	*514.50*
Tank-P4	*175.51*	*1.26*	*175.29*	*1.21*	*265.88*	*91.83*	*255.00*	*124.00*	*3,988.80*	*290.25*	*3,948.50*	*453.00*

IQR - Interquartile Range

Table 5-7: A comparison between the termination conditions between trials of both experiments.

Model	*Benchmark Experiment Termination*		*New Termination Conditions Experiment*			*New Termination Breakdown*	
	Original	*Time*	*Original*	*New*	*Time*	*Repeating solutions*	*1% of BTL*
2x2m	*100*	*0*	*57*	*43*	*0*	*41*	*2*
6x6m	*98*	*2*	*3*	*97*	*0*	*14*	*83*
House	*100*	*0*	*25*	*75*	*0*	*65*	*10*
House-W	*100*	*0*	*4*	*96*	*0*	*85*	*11*
Tank	*86*	*14*	*1*	*99*	*0*	*10*	*89*
*Tank-P4**	*5*	*45*	*0*	*100*	*0*	*0*	*100*

**Only 50 trials where conduced for the benchmark experiment*
BTL – Best tour length

90% of trials in the benchmark experiment terminate due to the imposed time limit, successfully completed all 100 trials. An analysis of Table 5-7 indicates that *smaller planning problems* generally terminated due to non-consecutive repeating solutions while trials of the *larger planning problems* terminated due to generating solutions that contribute no further improvement. As these termination conditions were triggered across all planning problems, the flexibility of these new conditions to terminate on behaviour rather than on set size was clearly evident.

A relative comparison between the trials in the benchmark experiment and the trials using the *additional termination conditions* have shown a reduction in overall planning times and have maintained quality in tour length for all planning problems (Table 5-8). Removing the *state of minimal improvement* reduced the number of planning iterations in the *small planning problems* by 6% – 17% ($p < 0.001$; Table 5-9) and by 77%– 98% ($p < 0.001$) in the *large planning problems*.

The results in Table 5-10 shows that fewer iterations reported ZEIs. The largest was *6x6m* at 24% but overall, the number of total iterations had reduced from 5,227 to 38 where over 95% of the 5,227 iterations were ZEIs. Since the *additional termination conditions* detected non-consecutive repeating solutions, fewer iterations returned repeating solutions. At most, the solution may repeat three times, but this was due to the termination conditions tracking if the repeating solution was less than the previous solution. As a result, the LPP produced meaningful solutions per iteration as more planning iterations are unique solutions. With the variability significantly reduced across all planning problems, these results better reflect how much time these planning problems should take to solve given the constraints applied.

Table 5-8: Relative performance and percentage difference between trials using the original and *additional termination conditions.*

Model	*Overall Time*		*CSP Time*		*MPP Time*		*Tour Length*		*Iterations*		*Path Evaluations*	
	RP	*PD (%)*	*RP*	*PD (%)*	*RP*	*PD (%)*	*RP*	*PD (%)*	*RP*	*PD (%)*	*RP*	*PD (%)*
2x2m	*0.96*	*-3.79*	*0.999*	*-0.12*	*0.84*	*-16.34*	*1.0000*	*0.00*	*0.83*	*-16.73*	*0.99*	*-0.72*
6x6m	*0.16*	*-84.01*	*1.002*	*0.21*	*0.11*	*-89.38*	*0.9997*	*-0.03*	*0.11*	*-89.45*	*0.84*	*-15.83*
House	*0.88*	*-12.45*	*1.000*	*-0.01*	*0.85*	*-14.88*	*1.0000*	*0.00*	*0.83*	*-16.57*	*0.99*	*-0.89*
House-W	*0.92*	*-8.11*	*1.003*	*0.29*	*0.92*	*-8.11*	*0.9994*	*-0.06*	*0.93*	*-6.69*	*0.95*	*-5.28*
Tank	*0.24*	*-76.30*	*1.001*	*0.05*	*0.23*	*-77.22*	*0.9996*	*-0.04*	*0.23*	*-77.38*	*0.80*	*-20.13*
*Tank-P4**	*0.02*	*-98.04*	*1.002*	*0.20*	*0.02*	*-98.36*	*0.9992*	*-0.08*	*0.02*	*-98.39*	*0.69*	*-31.20*

**Only the first 50 trials are compared*
CSP - Coverage Sampling Problem
MPP - Multi-goal Planning Problem
RP - Average of relative performance between each trial s.t. mean(New./Old)
*PD - Average of percentage differences between each trial s.t. mean((New-Old)./Old)*100*

Table 5-9: Statistical and practical significance between trials using the original and *additional termination conditions.*

Model	*Overall Time^*		*CSP Time#*		*MPP Time^*		*Tour Length#*		*Iterations^*		*Path Evaluations^*	
	p	*d*	*p*	*d*	*p*	*d*	*p*	*d*	*p*	*d*	*p*	*d*
2x2m	*< 0.001*	*0.65*	*0.22*	*-*	*< 0.001*	*0.67*	*0.73*	*-*	*< 0.001*	*0.69*	*0.24*	*-*
6x6m	*< 0.001*	*1.25*	*0.08*	*-*	*< 0.001*	*1.25*	*< 0.001*	*0.04*	*< 0.001*	*1.25*	*< 0.001*	*2.16*
House	*< 0.001*	*0.39*	*0.84*	*-*	*< 0.001*	*0.39*	*0.79*	*-*	*< 0.001*	*0.43*	*0.01*	*0.08*
House-W	*< 0.001*	*0.42*	*0.003*	*0.25*	*< 0.001*	*0.42*	*< 0.001*	*0.04*	*< 0.001*	*0.41*	*< 0.001*	*0.33*
Tank	*< 0.001*	*1.66*	*0.56*	*-*	*< 0.001*	*1.66*	*0.05*	*0.05*	*< 0.001*	*1.66*	*< 0.001*	*1.78*
*Tank-P4**	*< 0.001*	*14.85*	*0.02*	*0.65*	*< 0.001*	*14.85*	*< 0.001*	*0.47*	*< 0.001*	*14.86*	*< 0.001*	*4.44*

**Only the first 50 trials are compared*
CSP - Coverage Sampling Problem
MPP - Multi-goal Planning Problem
^p - Sign test ($\alpha < 0.01$)
#p - Paired Samples T-test ($\alpha < 0.01$)
d - Cohen's d effect size

Table 5-10: Lazy point-to-point planning data for all planning problems using the *additional termination conditions*.

Model	*Average IT to solve problem**	*Average number of IT with ZEs as a % of total IT (%)^*	*Average number of IT after the FZEI as a % of total IT (%)^*	*Average number of PE after the FZEI as a % of total PE (%)^*	*Average number of repetitions as a % of total IT (%)^*	*Average number of repeated solutions**	*Average number of unique solutions**
2x2m	*2.90*	*14.46*	*14.68*	*0.36*	*18.26*	*0.63*	*1.61*
6x6m	*31.44*	*24.72*	*58.01*	*5.90*	*1.95*	*0.38*	*31.03*
House	*37.86*	*7.25*	*10.17*	*0.63*	*3.75*	*1.04*	*36.46*
House-W	*10,657.31*	*0.14*	*4.71*	*4.20*	*0.05*	*2.95*	*10,652.89*
Tank	*319.78*	*6.95*	*46.55*	*8.72*	*0.20*	*0.45*	*319.26*
Tank-P4	*265.88*	*8.09*	*49.89*	*7.45*	*0.00*	*0.00*	*265.88*

**Average taken over all trials*
^Average of the relative difference between values within each trial
ZE - Zero path Evaluation
FZEI - First Zero path Evaluation Iteration
IT - Iterations
PE - Path Evaluations

Reducing the number of planning iterations consequently reduced the number of path evaluations. For the *smaller planning problems*, path evaluations reduced by 0.5% to 5.5%. For *2x2m* this was not statically significant ($p = 0.24$) but was for *House* and *House-W* it was a significant ($p < 0.01$). For the *large planning problems*, path evaluations reduced by 15% to 31% ($p < 0.001$). The reduction of the number of planning iterations and path evaluations iterations resulted in MPP times being reduced across all planning problems. As a result, *small planning problems* solved the MPP 3% to 12.5% faster than before ($p < 0.001$) while the *large planning problems* reported a 76% to 98% ($p < 0.001$) improvement.

The reduction of the MPP time significantly reduced overall planning times across all planning problems ($p < 0.001$). Removing the variability enabled all planning problems to be solved consistently. Large planning problems *Tank* and *Tank-P,* which originally has solutions solved in two to six hours respectively, are now solved in just over seven minutes. A remarkable improvement that produces more reasonable planning times that are appropriate for online planning given the simplified constraints applied to the planning problem. *6x6m* also received a significant improvement now solving in 38 seconds instead of 45 minutes. *House-W* only received an 8.11% improvement. Given that the variability was mostly removed from the LPP solutions, there are only five iterations of the 10,657 iterations that do not produce a unique solution suggesting that in the *House-W* problem the main influence is still the environment.

Finally, the *additional termination conditions* have *allowed the LPP to terminate earlier without sacrificing tour quality*. The largest difference in tour length from the benchmark experiment was 0.8% over all planning problems. While planning problems *6x6m, House-W,* and *Tank-P4* all reported a statistically significant result ($p < 0.001$), only *Tank-P4* had a practically significant improvement ($d = 0.4$) which corresponded to the 0.8% reduction in path length. The minor reductions in tour length were due to supplying the BTL upon termination, a condition that was not guaranteed under the original LPP implementation.

5.6.2 Discussion

The analysis of the new benchmark experiment has shown that;

1) The *additional termination conditions* have accounted for the variability between TSP solutions. Implementing *additional termination conditions* to track the behaviour of the LPP has helped to stabilise the solution times across all planning problems.

2) Tracking and terminating the LPP when no significant improvements are being made ensures the LPP no longer terminates at random. As a result, a considerable number of planning iterations that contributed no improvement, have been removed from the planning solutions resulting in the significant reduction of MPP and overall planning times. Compensating for the TSP variability, especially in larger planning problems, allows the LPP to effectively use an approximation TSP solver when solving the MPP online.
3) Employing the *additional termination conditions* has allowed the LPP to reliably solve larger MPPs, which suffered significantly due to variability in the TSP solutions. Removing the possibility that the LPP will enter a *state of minimal improvement* allowed planning problems, *Tank* and *Tank-P4*, where the majority of trials terminated due to a six-hour time limit, to be solved consistently within 7.5 minutes.

There are limitations to using these *additional terminating conditions*. It was initially assumed that *2x2m* and *6x6m* would terminate in two iterations because no corrections to the initial *Euclidean assumptions* were required. However, due to the variability that was present, and the additional time given by the *additional termination conditions* to finding an alternative solution, it will not be possible to solve problems like *2x2m* and *6x6m* in two iterations.

Without sufficient knowledge of the environment and the connectivity between the configurations, it is difficult to determine, before solving the MPP, that the initial *Euclidean assumption* placed between these configurations is correct. Additional iterations, beyond the FZEI, will always be required when using the *additional termination condition*s to ensure the LPP does not terminate prematurely.

The results also suggest that terminating upon the FZEI is likely to provide a solution that would be within 1% of the BTL (Table 5-10). Table 5-10 highlights that for the *larger planning problem*s, upwards of 46% to 58% of the iterations after the FZEI can still cycle through solutions attempting to find a better solution. While it could be argued that MPP times could be further reduced if the LPP is terminated upon the FZEI, this goes against the premise that relaxing the termination condition allows the LPP the flexibility to solve problems until no improvements can be found. While a small percentage of iterations are still evaluating ZEIs, the results have been significantly reduced to more reasonable times

that the extra iterations are no longer the main concern.

In summary, the *additional termination conditions* have successfully reduced planning times and allowed larger planning problems to be solved within a reasonable time. Relaxing the initial termination condition by applying *additional termination conditions* to compensate for the variability of the approximation TSP solver has enabled these TSP solvers to be used to solve large MPPs while ensuring the best solution found is always provided.

5.7 Analysis of the LPP Planning Data for Online Implementation

Resolving the variability of planning times from the planning solutions ensures the LPP provides reliable solutions when run online. However, while the a*dditional termination conditions* have had a significant impact in reducing planning times, some concerns remain around its applicability to be used online. The LPP still dominates overall planning times. For *Tank* and *Tank-P4* the MPP accounts for 70% of the overall planning times and for *House-W*, solving the MPP accounts for 99.8%. A closer analysis of the planning data highlights two factors that contribute to this behaviour;

1) there are a significant number of path evaluations needed to solve a given planning problem, and
2) small changes within the environment can have a significant impact on MPP times.

5.7.1 Path Evaluations and Environmental Changes

The number of path evaluations required to solve a planning problem is relatively high in comparison to others (Table 5-6). Some planning problems evaluate 2 to 3 times more paths than what is required to form a solution. This is an expected behaviour of the LPP as it lazily evaluates new paths to find a better solution. Therefore, it should be expected that a significant proportion of the paths evaluated will not be included in the final tour. However, if path planning is to be expensive, evaluating a significant number of paths that will not be retained in the final solution will only increase overall planning times. Given the current implementation of the LPP, the analysis in Section 5.3 uncovered that the majority of the path evaluations for all planning problems occur before the FZEI and therefore cannot be removed by the *additional termination conditions*.

As discussed in Section 4.2.5, there are two types of paths that can be evaluated;

1) a direct straight-line path between two configurations, or
2) an RRT path.

If a direct path cannot be formed between two configurations, an RRT is called to resolve the planning query. Solving a path using an RRT is more expensive than resolving the validity of a direct path as more collision checks are required. In reality, if a robot model that represents a multi-legged robot, all paths would be resolved using an RRT and this would increase planning times. Given that RRTs are only used in this context to correct invalid direct paths, the number of RRTs evaluated in a given planning problem provides insight into how influential the environment is on the LPP.

A statistical analysis of the number and types of paths evaluated along with paths used to form the final tour is presented in Figure 5-5 and Table 5-11. As expected, obstacle-filled environments, *House, House-W, Tank* and *Tank-P4,* all required RRTs to resolve the incorrect path assumptions. For *House*, *Tank* and *Tank-P4*, RRTs accounted for 17%-35% of the total paths evaluated. *Path evaluations for these planning problems are largely dominated by direct paths.* This finding explains why the planning times for *Tank* and *Tank-P4* was within minutes and seconds for *House*. However, the influence the environment had on the LPP was evident in the results for *House-W*.

Between *House* and *House-W* a small geometric change resulted in *House-W* taking 606 times longer to solve. Given the results, *House-W,* despite its relatively small covering set sizes, had the largest computational time (1.9 hours), number of planning iterations (10,657) and path evaluations (12,717) of all the planning problems. Overall, *House-W* still evaluated 24% of the total paths in the planning problem, a reduction of 1.5% after the new termination conditions had been applied.

The significant increase in planning time between *House* and *House-W* was attributed to the number of RRT evaluations required to solve the planning problem. More specifically, as 92% of the 12,717 paths evaluated were RRT solutions suggests that several thousands of paths were further apart than initially estimated by the *Euclidean assumption*. Given all other extrinsic factors have been minimised in the planning problem, these results highlight the impact the environment has on the LPP.

Of the 11,906 evaluated RRTs for *House-W*, the final solutions, on average, only contain 2 to 3 RRT paths (0.02%). For all the computational effort expended in producing a solution, the final plan used very few RRT paths. This observation is present amongst all obstacle- filled environments tested in these experiments as 99% of paths within the final tours were direct paths. Due to the compactness of the covering sets, this finding comes as

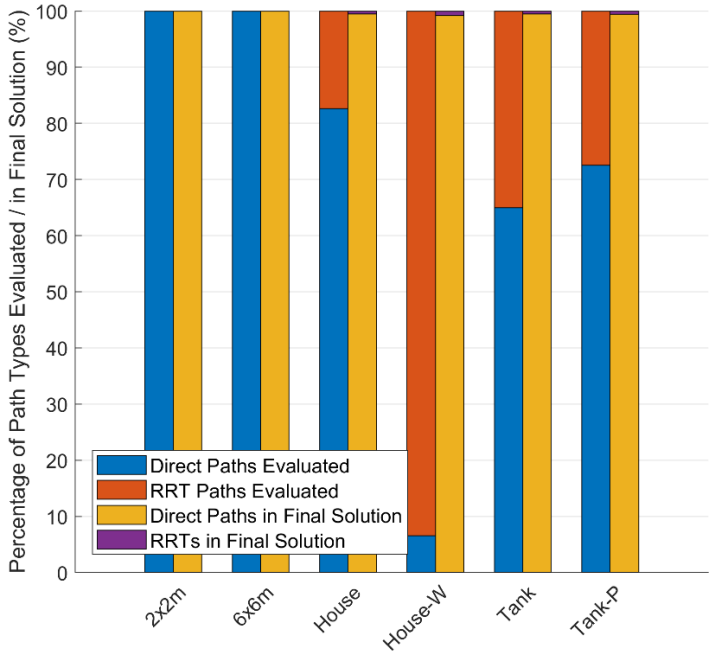


Figure 5-5: There are a significant number of RRT paths being evaluated that are not retained in the final solution.

Table 5-11: Analysis of the number of path types evaluated, taken and retained in the final solution.

Model	*Average # of paths in final solution*	*Average # of paths evaluated*	*Evaluated Path Breakdown*				*Final Plan Path Breakdown*				*Path Retention Rates*		
			Direct (Count / %)		*RRT (Count / %)*		*Direct (Count / %)*		*RRT (Count / %)*		*Direct (%)*	*RRT (%)*	*Combined (%)*
2x2m	*152.00*	*156.40*	*156.40*	*100*	*0.00*	*0.00*	*152.00*	*100*	*0.00*	*0.00*	*97.19*	*0.00*	*97.19*
6x6m	*1,011.05*	*1,363.76*	*1,363.76*	*100*	*0.00*	*0.00*	*1,011.05*	*100*	*0.00*	*0.00*	*74.14*	*0.00*	*74.14*
House	*321.80*	*557.62*	*456.99*	*82.61*	*100.63*	*17.39*	*320.20*	*99.50*	*1.60*	*0.50*	*70.07*	*1.59*	*57.71*
House-W	*325.35*	*12,717.84*	*811.82*	*6.54*	*11,906.02*	*93.46*	*322.62*	*99.16*	*2.73*	*0.84*	*39.74*	*0.02*	*2.56*
Tank	*1,301.44*	*3,594.35*	*2,322.32*	*64.97*	*1,272.03*	*35.03*	*1,295.00*	*99.51*	*6.44*	*0.49*	*55.76*	*0.51*	*36.21*
Tank-P4	*1,649.00*	*3,993.89*	*2,885.17*	*72.51*	*1,108.72*	*27.49*	*1,639.35*	*99.41*	*9.65*	*0.59*	*56.82*	*0.87*	*41.29*

no surprise. However, considering the number of RRTs that were required to obtain these solutions, reducing these evaluations will consequently reduce overall planning times further.

5.7.2 Impact on Online Replanning Strategies

In relation to the proposed online replanning strategies presented in *Chapter 3,* it would be expected that implementing a *full replan strategy* would suffer greatly under the current implementation of the LPP as more paths would be required to be evaluated compared to a *plan repair strategy*. As it is unknown as to when, where and what impact any new changes will have on the environment, and consequently on the coverage planner, replanning times could become quite volatile. Given the impact a singular geometric change had on the planning times between *House* and *House-W,* having equivalent replanning times that could increase from seconds to hours, due to environmental changes, would not be a desirable outcome from an online planner.

To be a more effective *adaptive coverage planner*, requires a better representation of the environment, encoded into the LPP. This improvement will reduce the number of path evaluations required to solve a planning problem and subsequently reduce overall planning times. The next chapter explores the path retention problem in further detail, providing an explanation of the behaviour and presents a novel heuristic that considers environmental influences to enable the LPP to plan more efficiently.

5.8 Chapter Summary

The benchmark experiment conducted in *Chapter 4* demonstrated that the LPP exhibited a variability that reduced the probability that the same covering set would produce similar planning times. The larger the covering set, the more significant the difference between planning times became. In this chapter, the planning data of the LPP was analysed to determine the cause behind the variable solution times.

The analysis of the LPP data determined that the variability observed by the LPP was due to the use of an approximation TSP solver coupled with the strict *equality termination condition* placed on the LPP. As there was no guarantee an approximation TSP solver would produce the same solution in successive iterations for an unchanged adjacency matrix, the termination of the LPP was left to chance. This resulted in many trials of the *larger planning problems* reaching the six-hour time limit imposed on a planning problem. If the LPP was to be used

in an online scenario, the variable solution times produced by the TSP solver needed to be resolved.

To compensate for the variable solutions, new termination conditions were added to track the behaviour of the LPP and terminate the planner if planning iterations did not meaningfully contribute to a better solution than what had currently been found. Rerunning the benchmark experiment showed the *additional termination conditions* were capable of stabilising the variability within the LPP resulting in reduced and consistent planning times.

Finally, the analysis of the new planning data highlighted that the MPP still dominates planning times. Despite the source of significant variability has been removed, the LPP still exhibits;

1) vulnerability to small changes within the environment can have a significant impact on MPP times, and
2) evaluates a significant number of paths to solve a given planning problem for which most are not retained in the final solution.

For an efficient online LPP implementation, these issues need to be addressed. Addressing these issues of the *offline sampling-based coverage planner* ensuring they are compensated for in an online implementation is investigated in the next chapter.

Chapter 6
Factoring Environmental Influences for Informed Path Planning

6.1 Introduction

Following on from the findings in *Chapter 5*, this chapter addresses the concerns about the efficiency of the *lazy point-to-point planner* (LPP) due to environmental changes. After compensating for the variability of the approximation *Travelling Salesman Problem* (TSP) solver with the *additional termination conditions*, the results of the *offline sampling-based coverage planner* were analysed. Two noticeable behaviours were present in the LPP data that allowed it to continue dominating planning times. The behaviours raised were;

1) small changes within the environment had a significant impact on solution times, and
2) a significant number of path evaluations were needed to solve a given planning problem that consequently lead to a significant number of evaluated paths not being retained in the final solution.

As there were minimal robotic constraints placed on the planning problem, the concerns were a direct consequence of the inadequate comprehension of the environment by the coverage planner. Given that, when replanning online it is unknown as to how, when or where changes will occur in the environment, giving the coverage planner a better metric, other than the *Euclidean assumption* to represent the environment, would ensure the LPP can efficiently provide tour updates when replanning online.

This chapter continues to investigate the LPP to determine if a more informative representation of the environment, encoded into the LPP, will reduce the number of path evaluations required to solve a given planning problem. This chapter explains why the LPP

evaluates so many paths, then a new heuristic is presented that improves the efficiency of the LPP by that factoring in the environment.

6.2 Underestimating the Connectivity between Configurations

Both concerns stated above can be linked to the initial *Euclidean assumption* placed on the connectivity between the configurations. As the coverage planner has no inherit understanding of either the environment or the connectivity between configurations, the connectivity is initially assumed to be Euclidean. From this assumption, the LPP is guided to construct a solution from the bottom up. Information about the environment is inferred upon each path evaluation, by the *rapidly exploring random tree* (RRT), as the LPP iterates towards the 'shortest' tour before terminating. The initial assumption ensures that any path evaluated by a motion planner will always be either equal or longer in length than the *Euclidean assumption* (Equation 6-1).

$$Euclidean\ Distance \leq Actual\ Distance \tag{6-1}$$

The shortest path assumption on the connectivity between configurations results in the underestimation of several paths within the environment. This underestimation is responsible for the increase in the number of path evaluations and low rates of retained paths in the final solution as demonstrated in the results for *House-W* (Section 5.7.1). However, if it was not for this basic assumption about the connectivity, the LPP would not be able to incrementally achieve quasi-optimal solutions.

To the LPP, *House-W* was the most difficult problem to solve, taking 1.9 hours to find a solution. In this time the LPP evaluated over 12,700 paths, of which 92% were RRTs (Figure 5-5 and Table 5-11). In Section 4.3.2 it was predicted that *House-W* would likely have higher computational times than *House* due to the additional paths required to resolve the underestimation around the change. However, the large increase in computational time from such a minor environment alteration was not expected. This is especially of note considering the representative tank environments contain covering sets 4-5 times the size of *House-W* yet solve significantly faster.

Figure 6-1 illustrates the evolution of a *House-W* LPP solution. It indicates that the early planning iterations produced a tour with little relation to the environment as many paths intersected the walls. Each of these intersections required an RRT to resolve the path query, thereby increasing the overall tour length. After 30 minutes, several thousand planning

iterations and path evaluations had occurred. Although the tour length at the 30-minute mark was within 6.9% of the final tour, the LPP continued searching for an additional 1.4 hours, evaluating on average 1.1 paths per iteration before stabilising on the final solution.

A simplified example of the underestimation for *House-W* is presented in Figure 6-2. Figure 6-2a shows a possible tour produced from the TSP solver for *House-W* starting and finishing in the same location (magenta). After evaluating all paths (green), one path (red) located at the additional wall, is required to be resolved by an RRT. The RRT-computed adjacency between these two configurations is now correctly represented but on the next iteration the selection of this path segment will have a lower priority due to other unevaluated paths that appear closer because of their *Euclidean assumption*. While the tour in Figure 6-2 is valid, it is unlikely that the same tour will be chosen by the TSP solver in the next planning iteration of the LPP, consequently preventing termination.

Figure 6-2b highlights all the potential path choices from the starting point that appear closer due to the *Euclidean assumption*. Many of these paths require the distance to be corrected by an RRT and consequently decreasing the likelihood of selection on subsequent iterations and in the final tour. As there are many candidates within close proximity, all but one new path may be evaluated per iteration. As it takes just one path to vary from the previous iteration for the termination conditions to fail, the LPP will continue evaluating paths, preferring underestimated paths, until a viable set of paths becomes present for the LPP to settle on a solution. Figure 6-2 highlights the impact that underestimation has on one configuration in the planning problem. In *House-W*, there are hundreds of configurations in the planning problem, and many of them suffer the same issue of having to resolve several underestimated paths before the TSP solver can stabilise to a solution.

House-W demonstrates that it is not necessarily the number of paths that are underestimated by the *Euclidean assumption* but how many paths must be evaluated to solve the *multi-goal planning problem* (MPP). These paths cannot be determined prior to initiating the LPP without an understanding of the environment. A suitable heuristic that can infer the properties of the environment to provide a better estimation of the connectivity will improve the efficiency of the LPP, as fewer path evaluations would be required. The closer the estimation is to the actual connectivity between the configurations, the faster the LPP will converge to a solution (Figure 6-2c).

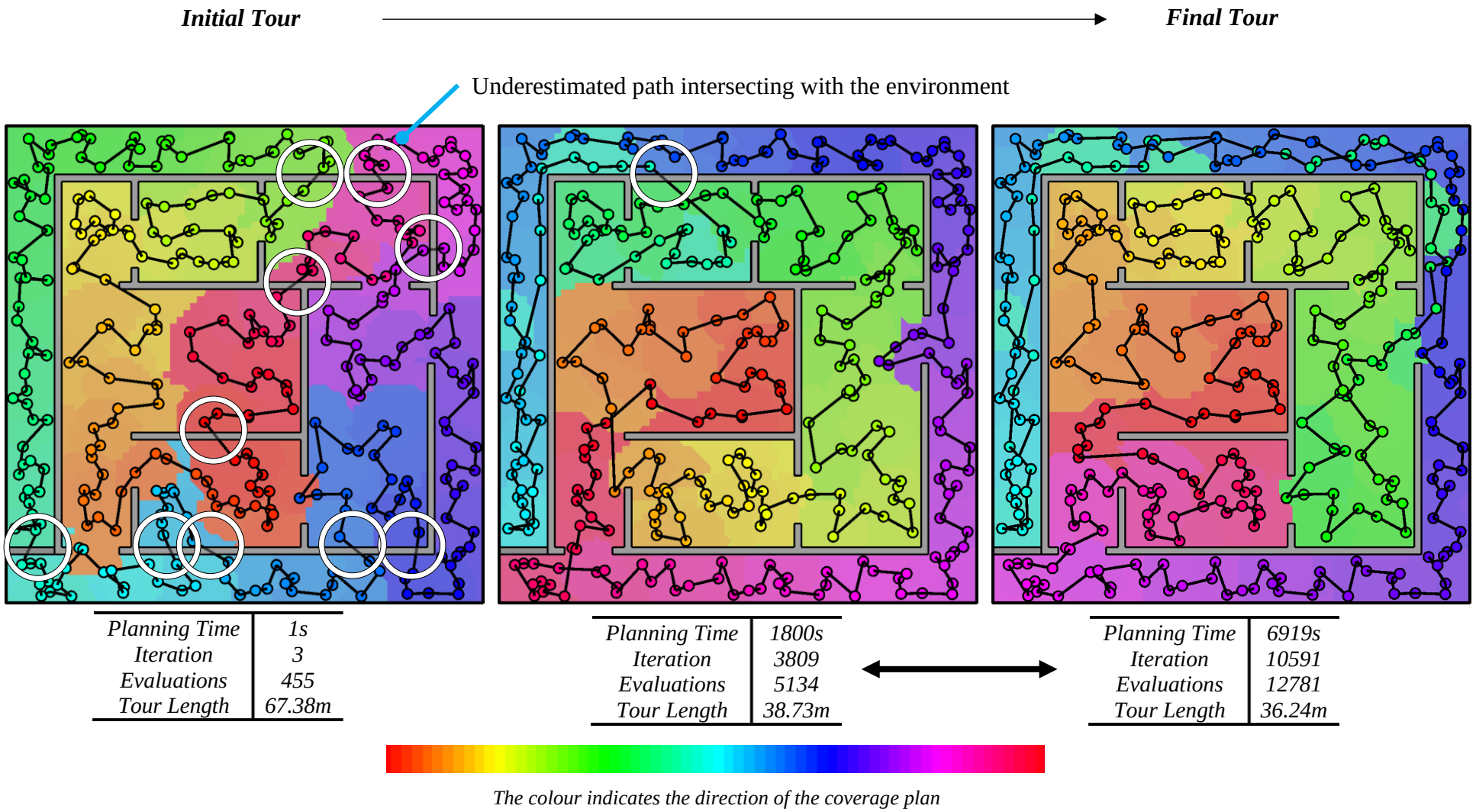
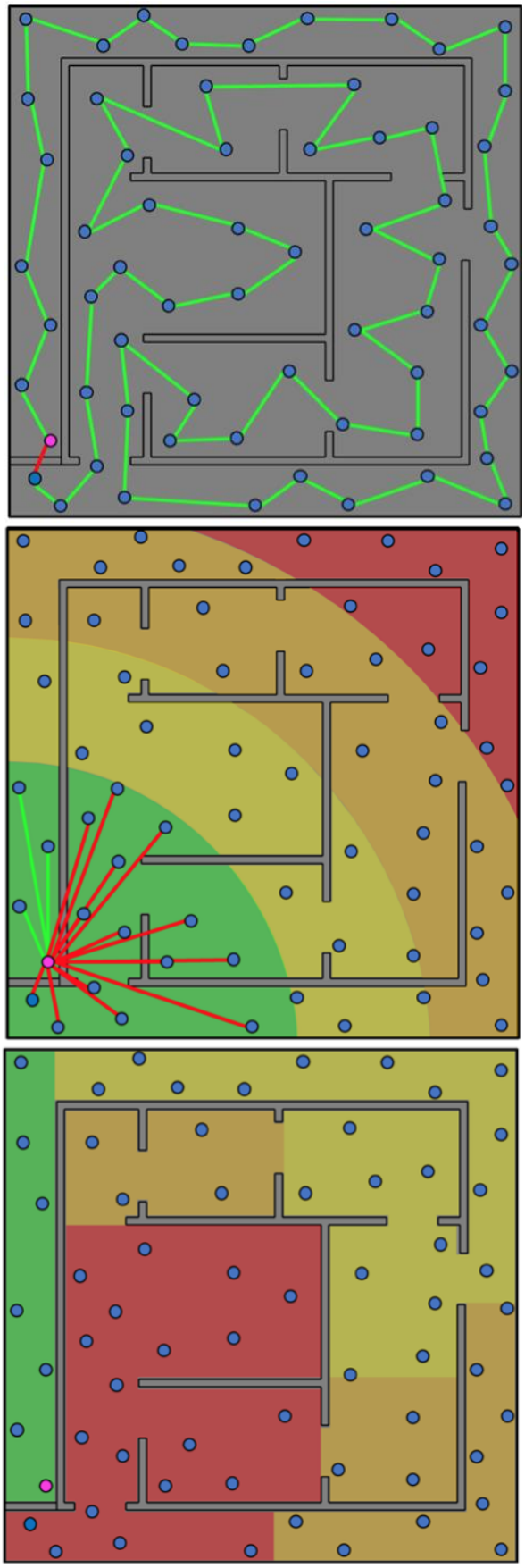


Figure 6-1: The evolution of a *lazy point-to-point planner* solution for *House-W*.



(a) A possible shortest path tour with one invalid path (red) to the start and finish position (magenta).

(b) The *Euclidean assumption* makes many 'local' configurations appear closer to the start position than what they actually are, resulting in many additional path evaluations.

(c) From the perspective of the start position, a distance will allow start position to route to local connections and remove the likelihood of connecting to the configurations on the other side of walls.

Figure 6-2: A simplified example demonstrating why the *Euclidean assumption* is an unsuitable metric in *House-W*. The colours in (b) and (c) represent the estimation of the connectivity from the perspective of the starting location.

6.3 Constructing a Suitable Heuristic

The number of path evaluations required to solve a planning problem can be reduced, if a heuristic that better estimates the connectivity between configurations, was applied before planning. Improving the estimation between configurations will remove paths that were initially considered to be close under the *Euclidean assumption* from the immediate solution space, thus providing the TSP with a better starting point to solve the MPP. By reducing the number of underestimated paths, it is expected that the LPP will converge faster to a solution. With fewer paths being evaluated, more evaluated paths will be retained in the final solution and that will consequently reduce overall planning times.

The *Euclidean assumption* is inexpensive to compute, so any heuristic that replaces it to encode information about environment will come at an additional cost. The use of a heuristic will be ineffective if the combined costs of calculating the heuristic and solving the MPP is equivalent to the cost of solving the MPP with a *Euclidean assumption*.

The time to compute a path between configurations is dependent on the complexity of the mobility constraints of the robotic platform and the complexity of the environment. Therefore, the calculation of the heuristic may not be applicable for all types of planning scenarios. If motion planning constraints and environments are simple, the cost of calculating the heuristic may outweigh the benefit of applying the heuristic. The heuristic's extended goal is for use in complex environments that would benefit from a reduction in the number of path evaluations used to solve a problem. A suitable heuristic must subsidise the cost of computing the heuristic to save time evaluating fewer paths, consequently reducing overall planning times. A suitable heuristic also needs to be robust enough to process a variety of enclosed environments and covering set sizes. As configurations are sampled at random, the heuristic needs to be applied after sampling and before path planning to avoid coupling the two processes.

Given the requirements placed on the heuristic to successfully replace the *Euclidean assumption*, it must satisfy the following criteria:

1) The heuristic must be calculated before solving the MPP.
2) It must reduce the time taken to solve the MPP by minimising the number of path evaluations required to solve a planning problem compared to a Euclidean solution.
3) It must not be a significant computational expense that increases overall planning times, outweighing the benefits of solving the MPP faster.

4) It must improve upon, or maintain, a similar tour quality to solutions solved under the *Euclidean assumption.*

Improving upon or maintaining similar tour quality is application specific and based on operator criteria. It could be argued that if there is a minor degradation in tour quality but planning times are significantly shorter, providing the longer tour length does not significantly increase execution time, the trade-off is acceptable. However, this trade-off is dependent on the application, robotic platform, and environment. As the immediate application of this research is for a multi-legged, high-DOF robot, all but the most negligible increases in tour length will proportionately increase execution time. It is therefore a priority to reduce the tour degradation, planning times and tour lengths, when using a new heuristic.

6.4 Extracting an Informed Graph through Environmental Properties

To gain a better estimation of the connectivity requires an understanding of the environment. Given that explicit parametrisation of the environment is not allowed (Section 1.5.1, *Requirement 3*), an approximation of the environment needs to be inferred before planning. This section discusses methods that encode the free-space connectivity of the environment into a graph without explicit parametrisation of the environment. The section will conclude by proposing a novel approach to path planning that has yet to be applied to the LPP.

6.4.1 Configuration-based Graphs

A simple extension to encode information about the environment into the adjacency matrix is to create a visibility graph between all configurations. Distances between configurations are generated by routing through line-of-sight neighbours. To find the shortest paths between all configurations an *all-pairs shortest path algorithm* such as the *Floyd-Warshall algorithm* (Floyd, 1962) can be used.

However, there are two limitations to this approach;

1) As the distance metric is based purely on the line-of-sight, there is no guarantee these distances represent the shortest of all possible routes through the environment. The graph may overestimate distances between configurations that are close together.
2) Line-of-sight needs to be guaranteed between all configurations to at least one other configuration, otherwise clusters of disconnected configurations will form. If clusters do form, another metric needs to be applied to connect these clusters; otherwise the adjacency matrix will be incomplete.

As the sampling of configurations is performed at random and the development of this coverage planner is to be used in complex geometries, line-of-sight between all configurations under these constraints cannot be guaranteed. More information about the environments is required to create a heuristic that is robust enough to ensure all configurations are connected and sufficiently estimated in complex environments.

6.4.2 Geometric and Topological Graphs that Represent Free Space

To obtain more information about the environment, the dimensionality of the environment may be represented as a 1D backbone using techniques such as the *medial axis transform* (Blum, 1967), a *Voronoi diagram*, and topological curve-skeletons. The result of these techniques is a graph that represents either the topological or the geometric connectivity of free space.

Despite the applicability of the *medial axis transform* and the *Voronoi diagram* in 2D, in 3D, the *medial axis transform* translates to a medial surface and *Voronoi* into a mesh vertex. Neither of these transforms are suitable to create either a graph or a distance metric in $\mathbb{R}^3$ without further processing (Cornea, Silver and Min, 2007; Tagliasacchi et al., 2016). *Topological skeletons*, on the other hand, do produce skeletons in $\mathbb{R}^3$ within one operation and are easily translated into a graph to facilitate their use in complex 3D environments.

Most approaches to skeletonisation incrementally erode a voxel representation of the environment, either objects or free space, until a thin 1D skeleton remains. Figure 6-3 illustrates an example of a 1D skeleton being formed from a topological thinning algorithm. As the skeleton is formed by eroding free space, it is known that any 1D skeleton will represent a series of collision-free paths. Such a representation of the environment has been shown to be useful when solving complex path planning problems, with examples including:

1) Routing efficient paths through sensor networks (Bruck, Gao and Jiang, 2007).
2) Collision-free path trajectories for robot motion planning in unknown environments (Thrun, 1998; Choset and Burdick, 2000).
3) High-level mission planning for autonomous surface vehicles (Wheare, 2018).

All these examples use a skeleton of the environment to inform a planner of the connectivity of the environment before planning. This indicates that a skeleton can be used to approximate connectivity between the configurations, and thus be a suitable heuristic for the LPP.

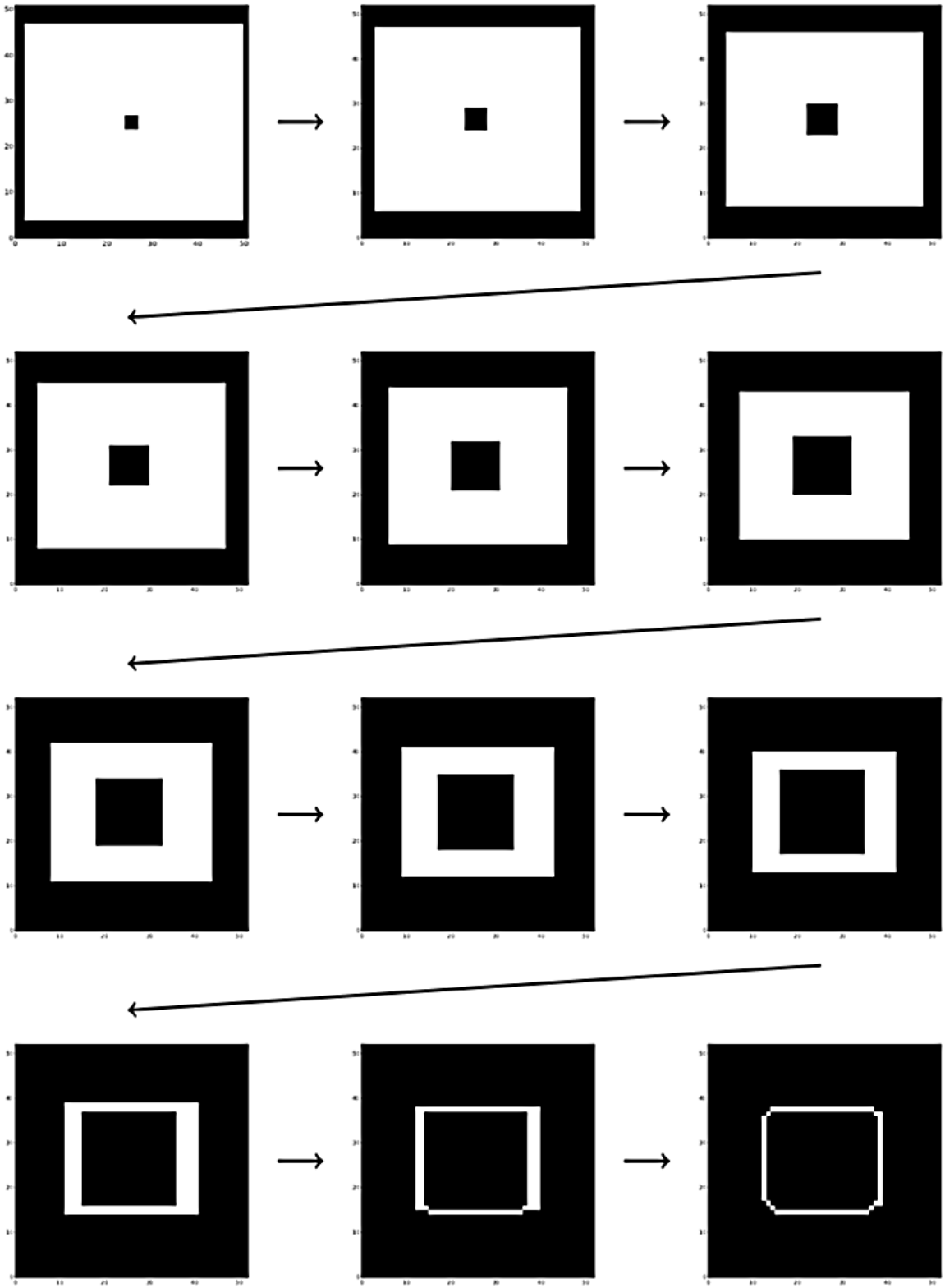


Figure 6-3: Evolution of the Lee et al. (1994) thinning algorithm producing a skeleton within a rectangular space around a central structure. Each image demonstrates how the space is iteratively eroded away from all surfaces (black) until a thin 1D skeleton remains between the sides and the centre object (Wheare, 2018, reproduced with permission).

The properties of topological and geometrical thinning algorithms have been well described in Cornea et al., (2007). An important property of topological skeletons is that they preserve connectivity. All topological thinning algorithms perform checks to ensure that the topology of the environment remains in the resultant skeleton. This property ensures that if the configurations of a covering set can be appropriately connected to the skeleton, the properties of a topological thinning algorithm guarantee all configurations are connected (Figure 6-4). Therefore, the estimated distances between configurations will be via the approximated collision-free paths of the skeleton, which will result in a better estimation of configuration connectivity.

6.4.3 Skeleton-heuristic Proposal

To resolve computational time issues stemming from the underestimation of the Euclidean assumption, it is proposed to use a topological thinning algorithm to create a 1D skeleton that will provide the LPP with an estimation of the connectivity before solving the MPP. A better estimate of connectivity between configurations will reduce the chance of evaluating non-viable paths, therefore resulting in a higher proportion of evaluated paths being selected for the final solution.

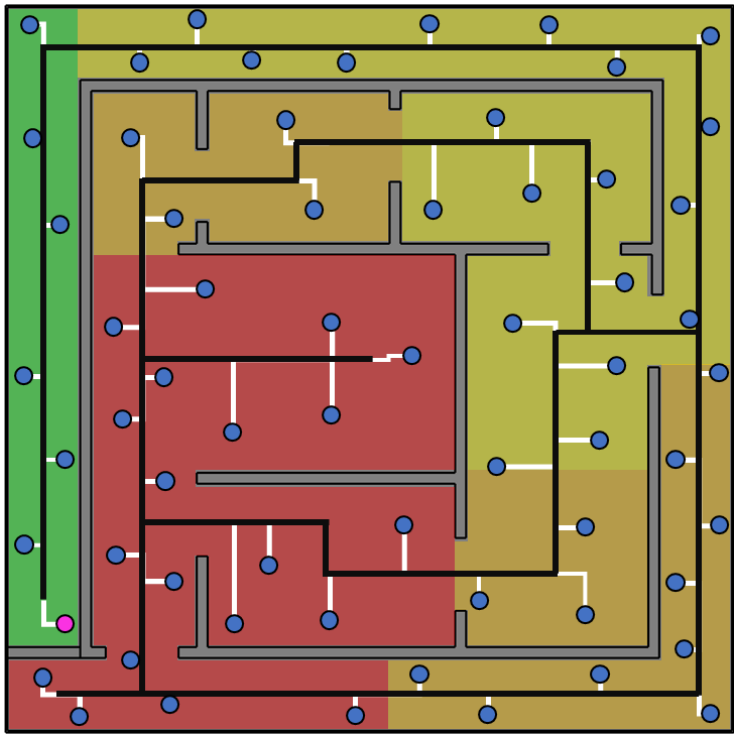


Figure 6-4: Using a skeleton of the environment will allow a graph to be formed that allows the connectivity between configurations to better represent the actual distance, thus aiding the removal of unnecessary path evaluations from occurring. The colours are used to represent the connectivity of the magenta configuration to the other configurations in the planning problem. Green represents a close connection and red represents a distant connection.

As the resultant skeleton approximates the actual collision-free paths the robot might take through the environment, the skeleton will overestimate distances between configurations. Since the connectivity between configurations is better approximated, paths that are further away, which may be overestimated, will be assigned a lower priority over paths that are close together. As the LPP underestimates all paths to build up to a solution, this overestimation of paths opposes the original implementation of the LPP. However, given that it is the underestimation that significantly impacts planning times in complex environments, the overestimation of distant paths is acceptable, providing the skeleton-heuristic can meet the criteria for a suitable heuristic (Section 6.3). To the author's knowledge, this approach has not been applied to an LPP in an attempt to reduce the number of path evaluations needed and the time is takes to produce a solution.

6.5 Constructing an Adjacency Matrix Using the Skeleton-heuristic

The binary-thinning algorithm used to achieve the proposal is Lee et al. (1994) *3D parallel thinning algorithm*. Lee's thinning algorithm is applicable as it produces a 1D skeleton of a 3D environment that preserves both topological and geometrical properties. Topological skeletons always ensure the topology of the environment is preserved, resulting in a skeleton that is fully connected.

The *parallel thinning algorithm* allows for more information to be held about the environment as its spatial information is reduced to a lower dimensionality. If a fully topological algorithm was used for the heuristic, it would erode the free space of narrow geometries that do not contain topological features (Cornea, 2007). If narrow compact spaces are dissolved and these areas contain configurations, using a fully topological algorithm, such as Palagyi and Kuba (1999), may decrease the chances of configurations being directly connected to the skeleton.

Conversely, fully geometrical thinning algorithms are known to create many branches as they are sensitive to deviations in the surface of the environment. Generally, these extra branches need to be pruned (Cornea, 2007), otherwise they could make graph searching prohibitively expensive. Therefore, Lee's thinning algorithm is a suitable candidate to test the skeleton-heuristic proposal. This thesis uses Homman's implementation of Lee's 3D thinning algorithm (Homman, 2007).

6.5.1 Connecting Configurations to the Skeleton during the Thinning Process

Homman's implementation of Lee et al. (1994) thinning process presents four rules that are used to determine if a voxel is removed or is to be a part of the skeleton. These cross-checks are applied iteratively over each voxel until a 1D skeleton is formed. A summary of the four rules for voxel deletion are:

Rule 1) The current voxel must be a surface voxel.
Rule 2) The voxel must not be at the end of a line.
Rule 3) The deletion of the voxel would not change the Euler characteristic.
Rule 4) The deletion of a simple voxel does not change the number of connected objects.

These four rules guarantee that the skeleton is fully connected upon completion. More information about the Euler characteristic and simple voxels, referred to as 'points' in 2D, can be found in Lee et al. (1994).

To construct a graph that includes the distances to all configurations, the configurations must be connected to the appropriate skeleton branches. Configurations can be connected via point to line, or by using a watershed segmentation (Mangan and Whitaker, 1999; Wheare, 2018). Either method provides an estimate as to where along the skeleton is the most appropriate place to connect the configurations. However, these approaches entail a two-step process that requires;

1) the calculation of the skeleton, and
2) the connection of the configurations to the skeleton.

In an attempt to remove a step between creating the skeleton and the adjacency matrix, an alternative approach is proposed. A fifth rule is added to the list of voxel deletion rules above,

Rule 5) A voxel must not contain the Cartesian position of a configuration.[3]

Rules 1-4 ensure that once a voxel is determined to be a part of the skeleton it will not be removed. Rule 5 ensures that each branch that begins at the Cartesian position of a configuration will remain connected to the final skeleton.

Figure 6-5 shows the resultant skeletons for *House-W* of Lee's thinning algorithm against a skeleton created with the amended fifth rule. The skeleton of Figure 6-5b is representative

[3] Rule 5 amendment courtesy of James Armitage

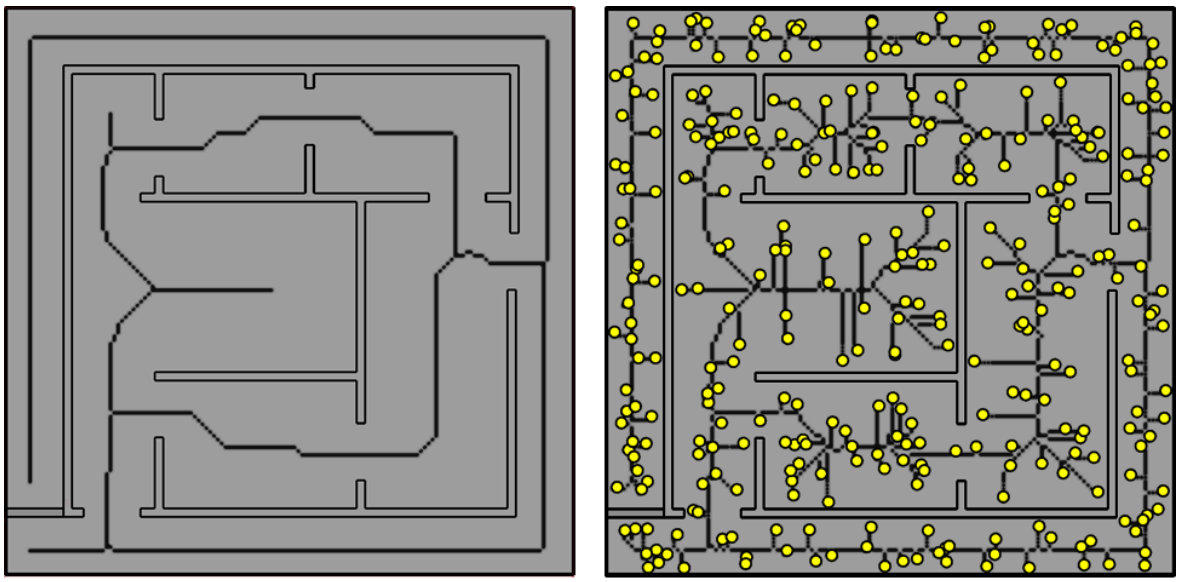


Figure 6-5: Resultant skeletons (black) of *House-W* using the (a) traditional Lee et al. thinning algorithm without the extra voxel deletion criteria and (b) connecting configurations (yellow) to the main skeleton with the extra voxel deletion criteria. As the walls are hollow, a skeleton is also generated within them. For clarity, these skeletons are removed from the image.

of the skeleton proposal in Figure 6-4. Since all configurations are connected into one continuous skeleton, an adjacency matrix that represents the connectivity between the configurations through the use of skeletons can be used to inform the LPP.

6.5.2 Computation of the Skeleton-heuristic

Implementing the skeleton-heuristic requires the mesh representation of the environment to be in a voxel format. Min's Binvox implementation of a mesh-to-voxel converter was used to transpose surface mesh models into a voxel format suitable for thinning (Min, 2017; Nooruddin and Turk, 2003). To minimise the voxelisation and skeleton computation, BinVox was also used to scale down the size of the environment to avoid thinning a high-resolution voxel model. Voxelised models were scaled to the desired resolution. i.e. a 15x20m mesh model that is voxelised to a 20mm resolution will result in a scaled down 750x1000 voxelised model.

As the model is scaled down, it is possible for two configurations to share the same voxel, in which case, the distance between the two configurations is assigned to be one voxel apart. Therefore, when the adjacency matrix has been formed, and the voxel distances are scaled to full size, the smallest possible distance between configurations will be the voxel resolution. As the smallest distance between two configurations is a voxel, when the distances are scaled back to the Euclidean space, no skeleton distance can be equal to the

Euclidean distance (Equation 6-2).

$$Euclidean\ Distance < Skeleton\ Distance \tag{6-2}$$

Further details on the algorithms and packages used to create a skeleton-heuristic can be found in Appendix A.

6.5.3 Limitations to the Proposed Skeleton-heuristic Implementation

There are few limitations to the proposed implementation:

1) Configurations cannot be contained within a boundary voxel.
2) Increasing the resolution increases computational time.
3) Narrow passageways that are not traversable by the robot can be included in the skeleton due to the resolution of the voxels, this creates a branch that will result in an underestimated connectivity in particular regions.

When rasterising a mesh to voxels, the quantisation limits the accuracy of the representation of the original mesh boundaries. Therefore, it is difficult to determine if the region within one voxel unit of the original boundary is a part of the voxelised area or a wall. Thus, if a configuration's coordinates are within one voxel unit of the original boundary, the configuration's voxel may not be part of the region that is skeletonised. If the configuration is not part of the skeleton it will be removed from the planning problem.

To ensure configurations are all in the correct area after voxelising, they should be at least one voxel unit from any boundary. Determining the correct resolution to voxelise complex environments is challenging and requires a manual process. As a guide, the resolution should be at least half the width of the collision model, or finer than the minimum viewing distance. Configurations with distances to the surface under these bounds will not be included in the final adjacency matrix and consequently will invalidate a complete coverage solution. Care needs to be taken to find the appropriate resolution for each planning environment. Due to $O(n^3)$ nature of the algorithm, increasing the resolution has a significant impact on computation time. In larger environments, especially with environments that have significant z-axis, the process will suffer from having a skeleton produced with a finer resolution.

Finally, there may be circumstances in which a skeleton will not be a suitable distance metric. Lee's thinning algorithm ensures the resultant skeleton will always be connected. If even a single line of empty voxels exists between two surfaces, a skeleton will be produced within this space. However, while the resolution of the voxels may capture the narrow

passageways, these passageways may not be physically possible for the robot to pass through the apparent collision-free path. As a result, these distances will reflect what movements can be achieved. However, in reality the distances will underestimate the connectivity derived through these skeleton branches in a similar fashion as the Euclidean assumption.

Since the skeleton still informs the TSP of the remaining connectivity of the space, in such cases when underestimation occurs due to the skeleton, it is expected that planning times will be longer but not as long as a Euclidean solution. A solution that could compensate for this issue would be to perform collision checks along the skeleton and update the adjacency appropriately for edges of the graph that cannot be achieved. Rerunning any shortest path algorithm will resolve the connectivity issue. However, this was not implemented in this work as all environments tested did not have this property.

6.6 Coverage Planning using the Skeleton-heuristic

To determine the effectiveness of the *skeleton-heuristic*, the offline benchmark experiment performed in Section 5.6, with the *additional termination conditions*, was repeated. A relative comparison between the trials of the termination condition experiment was carried out to determine if the new *skeleton-heuristic* improves upon solution times, planning iterations, path evaluations and corresponding impact on tour quality. A Paired Samples T-Test ($\alpha = 0.01$) was performed to determine whether a significant statistical (p) and practical (d) difference existed between trials.

As the skeleton-heuristic is introduced to improve the MPP times, the MPP times presented in Table 6-1 combine the time to solve the skeleton-heuristic and the time taken for the LPP to solve the MPP. Table 6-2 provides the analysis between these two processes. When comparing the relative performance and statistical significance between MPP times, both MPP times include the time taken to construct the initial condition of the LPP.

To allow for a fair and valid comparison between the trials of both experiments, all trials were conducted on the same 64-bit Intel i7 920 CPU, 8 core, 6GB RAM machine running Ubuntu 16.04 LTS as used in the benchmark experiment in *Chapters 4* and *5*. All the analysis was performed in MATLAB 2018b.

6.6.1 Computational Observations and Results

Solving the MPP with a *skeleton-heuristic* has significantly reduced the number of path

evaluations required to solve a given planning problem (Table 6-1). All planning problems recorded a reduction in path evaluations compared to planning problems solved with a *Euclidean assumption* ($p < 0.001$, Table 6-4). *House-W* witnessed a 97% reduction in path evaluations (Table 6-3), evaluating 12,361 fewer paths than the Euclidean trials.

Evaluating fewer paths caused the LPP to require fewer planning iterations to converge towards a solution. While all planning problems recorded a reduction of planning iterations, obstacle-filled environments *House*, *House-W*, *Tank*, and *Tank-P4*, benefited significantly from evaluating fewer paths as they all record at least an 86% reduction in planning iterations ($p < 0.001$). Non-obstacle filled environments *2x2m* and *6x6m* recorded smaller benefits of 5% and 25% respectively with *2x2m* recording no statistical significance ($p = 0.16$) as there was only a 1-2 iteration difference from the Euclidean results.

The significant reduction of planning iterations and path evaluations had a positive effect on MPP times. With the exception of *2x2m*, which did not receive an overall improvement in planning time due to the cost of calculating a skeleton, all other planning problems MPPs solved significantly faster than the planning problems which used the *Euclidean assumption* ($p < 0.001$). Under the *Euclidean assumption*, solving the MPP accounted for 90% of the overall planning time (Table 5-5). The introduction of the skeleton lowered the costs of solving the MPP, as the MPP times accounted for up to 50% of the overall planning time. For most planning problems, the calculation of the *skeleton-heuristic* became the most dominating process. As expected, thinning the voxelised environment was the most expensive process when creating the skeleton (Table 6-2).

In the case of *2x2m*, the combined time of creating the *skeleton-heuristic* and solving the MPP was double the amount of time of solving the same problem with the *Euclidean assumption*. For the other planning problems, solving the MPP with the *skeleton-heuristic* reported better overall planning times ($p < 0.001$), than the equivalent Euclidean trials (Table 6-3).

Obstacle-filled environments *House*, *House-W*, *Tank* and *Tank-P4* produced the greatest improvement, with MPP times being reduced by at least 63%. Remarkably, *House-W* recorded a 99.9% improvement. Coverage plans for *House-W* solved within 5.9 seconds using a *skeleton-heuristic*, as opposed to coverage plans using the *Euclidean assumption* that solved in just under two hours (Table 5-5). However, despite the improved planning times, solving the MPP with the *skeleton-heuristic* came at the expense of tour quality.

Table 6-1: Planning times, tour lengths and *lazy point-to-point planner* attributes for all planning scenarios using the *skeleton-heuristic*.

Model	*Trials*	*Overall Time (s)*		*MPP Time* (s)*		*Tour Length (m)*		*Iterations*		*Path Evaluations*		*Configurations*
		$\bar{x}$	*SD*	$\bar{x}$	*SD*	$\bar{x}$	*SD*	$\bar{x}$	*SD*	$\bar{x}$	*SD*	
2x2m	*100*	*5.55*	*0.05*	*3.04*	*0.03*	*45.34*	*2.04*	*2.70*	*0.73*	*152.00*	*2.96*	*152.00*
6x6m	*100*	*32.87*	*2.89*	*10.91*	*2.76*	*405.98*	*7.64*	*6.79*	*5.45*	*1,022.42*	*13.97*	*1,011.05*
House	*100*	*4.79*	*0.10*	*2.12*	*0.09*	*42.03*	*1.01*	*2.66*	*0.83*	*321.94*	*5.53*	*321.80*
House-W	*100*	*5.91*	*0.39*	*3.18*	*0.39*	*46.23*	*1.20*	*3.20*	*1.47*	*327.06*	*5.81*	*325.35*
Tank	*100*	*106.48*	*6.36*	*73.71*	*6.31*	*274.28*	*4.64*	*15.34*	*10.17*	*1,347.94*	*20.72*	*1,301.76*
Tank-P4	*100*	*155.61*	*9.19*	*89.28*	*9.19*	*326.68*	*5.13*	*33.16*	*13.52*	*1,797.23*	*46.65*	*1,649.00*

MPP - Multi-goal Planning Problem

**MPP Time includes the time to create the skeleton and distance metric. A breakdown of the skeleton and MPP solution time can be found in Table 6-2*

Table 6-2: Analysis of the time taken to create the *skeleton-heuristic* and the resultant time to solve the *multi-goal planning problem* (MPP).

Model	*MPP Time (s)*	*Calculating the Skeleton-heuristic*						*Solving the MPP*		
		Voxelisation			*Thinning*					
		$\bar{x}$ ($x10^{-3}$s)	*SD* ($x10^{-3}$s)	*TT (%)*	$\bar{x}$ *(s)*	*SD (s)*	*TT (%)*	*M (s)*	*SD (s)*	*TT (%)*
2x2m	*3.04*	*2.79*	*0.15*	*0.09*	*2.49*	*0.02*	*81.88*	*0.55*	*0.02*	*18.03*
6x6m	*10.91*	*11.73*	*0.86*	*0.11*	*7.17*	*0.10*	*68.86*	*3.73*	*2.78*	*31.03*
House	*2.12*	*2.37*	*0.14*	*0.11*	*1.59*	*0.04*	*75.17*	*0.53*	*0.08*	*24.72*
House-W	*3.18*	*2.34*	*0.08*	*0.07*	*1.58*	*0.04*	*50.23*	*1.60*	*0.38*	*49.70*
Tank	*73.71*	*9.76*	*0.67*	*0.01*	*63.50*	*0.30*	*86.70*	*10.20*	*6.33*	*13.29*
Tank-P4	*89.28*	*13.59*	*0.97*	*0.02*	*66.59*	*0.37*	*75.31*	*22.68*	*9.11*	*24.67*

TT - Time Taken as a percentage of MPP time

Table 6-3: Relative performance and percentage difference between trials using the *Euclidean assumption* and the *skeleton-heuristic*.

Model	*Overall Time*		*MPP Time*		*Tour Length*		*Iterations*		*Path Evaluations*	
	RP	*PD %*	*RP*	*PD %*	*RP*	*PD %*	*RP*	*PD %*	*RP*	*PD %*
2x2m	*2.02*	*101.84*	*18.23*	*1,722.99*	*2.11*	*111.18*	*1.05*	*5.40*	*0.97*	*-2.58*
6x6m	*0.87*	*-12.64*	*0.81*	*-19.10*	*2.55*	*154.66*	*0.25*	*-74.85*	*0.75*	*-24.82*
House	*0.52*	*-48.40*	*0.36*	*-63.87*	*1.24*	*23.84*	*0.11*	*-89.10*	*0.59*	*-41.35*
House-W	*0.0009*	*-99.91*	*0.0005*	*-99.95*	*1.29*	*29.49*	*0.0003*	*-99.97*	*0.03*	*-97.35*
Tank	*0.27*	*-72.91*	*0.21*	*-79.34*	*1.84*	*84.16*	*0.05*	*-94.86*	*0.38*	*-62.22*
Tank-P4	*0.41*	*-58.88*	*0.29*	*-70.94*	*1.87*	*86.63*	*0.13*	*-86.59*	*0.45*	*-54.79*

RP - Average of relative performance between each trial s.t. mean(New./Old) *MPP - Multi-goal Planning Problem*
*PD - Average of percentage differences between each trial s.t. mean((New-Old)./Old)*100*

Table 6-4: Statistical and practical significance between trials using the *Euclidean assumption* and the *skeleton-heuristic*.

Model	*Overall Time*		*MPP Time*		*Tour Length*		*Iterations*		*Path Evaluations*	
	p	*d*	*p*	*d*	*p*	*d*	*p*	*D*	*p*	*d*
2x2m	*<0.001*	*37.58*	*<0.001*	*51.24*	*<0.001*	*20.61*	*0.16*	*-*	*<0.001*	*0.79*
6x6m	*<0.001*	*1.31*	*<0.001*	*1.30*	*<0.001*	*56.70*	*<0.001*	*3.00*	*<0.001*	*7.82*
House	*<0.001*	*2.46*	*<0.001*	*2.48*	*<0.001*	*10.27*	*<0.001*	*2.86*	*<0.001*	*6.08*
House-W	*<0.001*	*9.74*	*<0.001*	*9.74*	*<0.001*	*12.60*	*<0.001*	*11.74*	*<0.001*	*11.48*
Tank	*<0.001*	*4.96*	*<0.001*	*4.97*	*<0.001*	*43.81*	*<0.001*	*5.51*	*<0.001*	*13.59*
Tank-P4	*<0.001*	*4.09*	*<0.001*	*4.10*	*<0.001*	*47.12*	*<0.001*	*4.56*	*<0.001*	*12.77*

MPP - Multi-goal Planning Problem

Tour lengths using the *skeleton-heuristic* were significantly longer than solutions solved using a *Euclidean assumption* ($p < 0.001$, $d > 0.8$, Table 6-4). Tour lengths for non-obstacle filled environments *2x2m* and *6x6m* were over twice as long as the original tour, while obstacle-filled environments solutions vary. *House* and *House-W* are 23% to 30% longer while *Tank* and *Tank-P4* tour lengths increased upwards of 84% to 87%.

Figures 6-6 and 6-7 compare the resultant tours solved using Euclidean and skeleton based metrics for *2x2m* and *House-W* respectively. The tours solved using the *skeleton-heuristic* produced tours that were markedly different to tours solved using the *Euclidean assumption*. Path segments within areas where no obstacles were present are longer than their Euclidean counterparts. This is clearly witnessed in Figure 6-6 as the resultant tour generated for *2x2m*. These results highlight a serious issue with the estimation of the *skeleton-heuristic* within open areas.

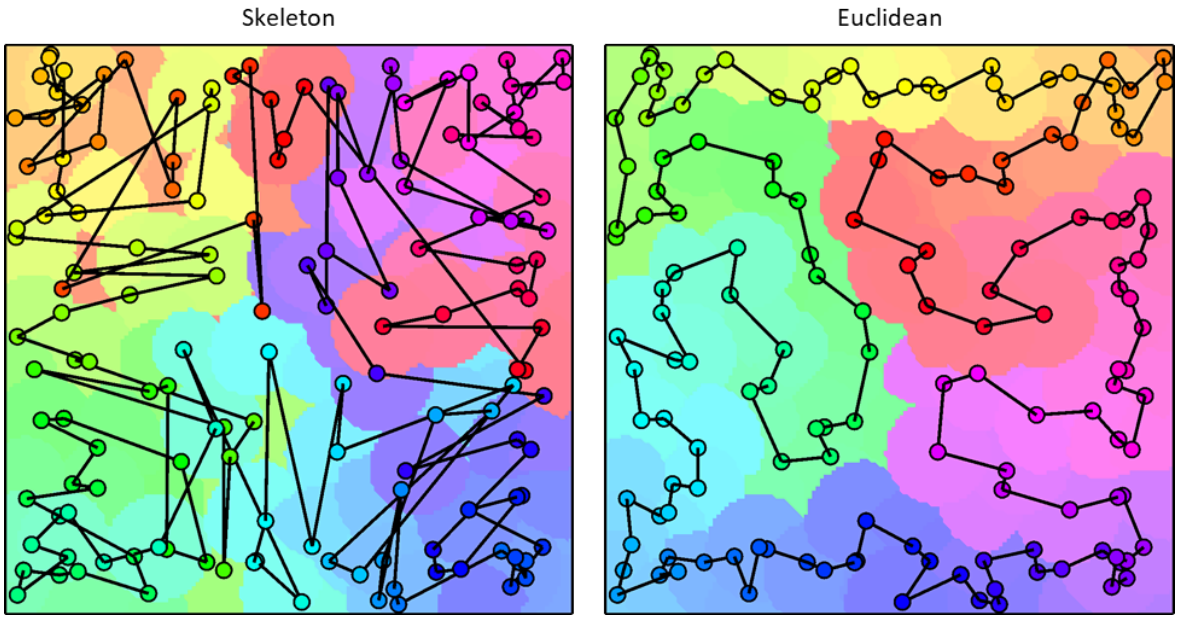
6.6.2 Discussion

The introduction of the *skeleton-heuristic* has successfully reduced the time it took to solve the MPP by reducing the number of path evaluations required to solve a given planning problem. The results show that the *skeleton-heuristic* reduced MPP times enough to subsidise the calculation of the *skeleton-heuristic* within the planning procedures. However, the reduction of the MPP times using the *skeleton-heuristic* came at a cost to the tour quality, therefore invalidating the *Criteria (4)* for a suitable replacement to the initial *Euclidean assumption*.

It could be argued that the 99.9% improvement in overall planning times for *House-W* is an acceptable trade-off for a 29.5% increase in tour length. However, for the representative tank models, a 58% to 72% reduction in planning times for an 84% to 87% increase in tour length may outweigh the costs of calculating a plan faster when executing the tour in reality. A clear argument against the use of the *skeleton-heuristic* is how ineffective it was for non-obstacle filled environments, *2x2m* and *6x6m*. The time to calculate the skeleton increased overall planning times for *2x2m* and increased tour length for both planning problems.

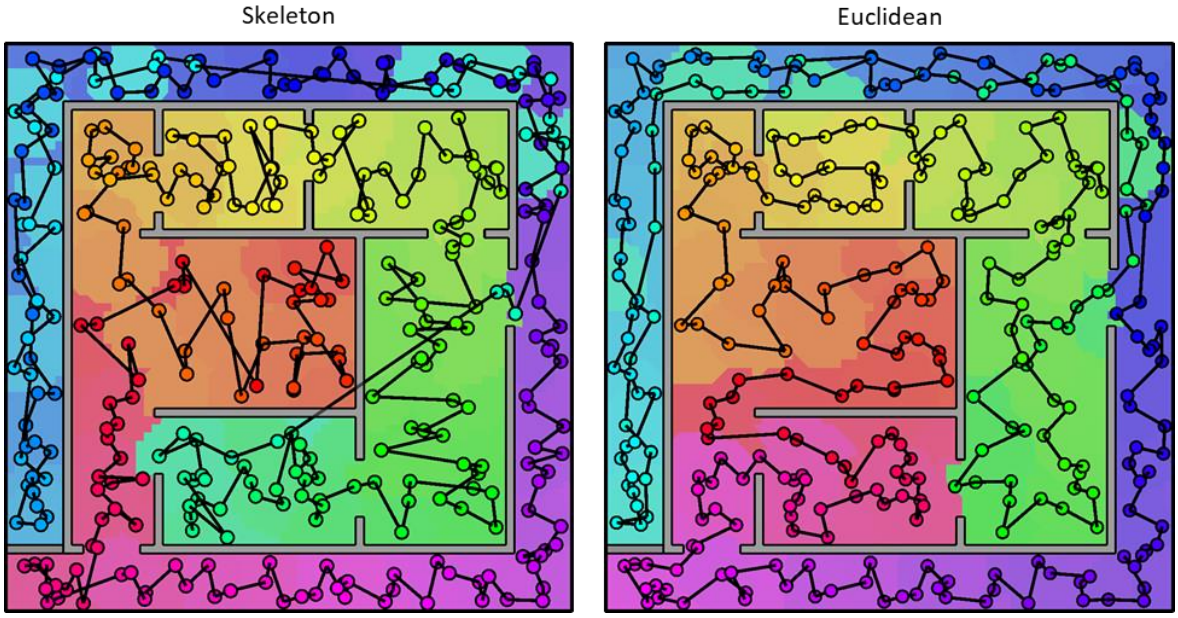
Ultimately, the trade-off between path length and the time to execute a coverage plan is;

1) subjective to the operator and task at hand,
2) the robotic platform used for the inspection task, and
3) the type of environment the plan is executed in.



2x2m	*Overall Time*	*Tour Length*
Skeleton	*5.55s*	*43.61m*
Euclidean	*2.67s*	*21.74m*
Percentage Difference	*107.87%*	*100.60%*

Figure 6-6: The final tour generated by the *lazy point-to-point planner* initialised with a *skeleton-heuristic* and *Euclidean assumption* for *2x2m*. The *skeleton-heuristic* increased planning times and tour length for *2x2m* due to the calculation of the heuristic and the overestimation of local connectivity respectively.



House-W	*Overall Time*	*Tour Length*
Skeleton	*5.91s*	*45.68m*
Euclidean	*4028.91s*	*36.03m*
Percentage Difference	*-99.85%*	*26.78%*

Figure 6-7: The final tour generated by the *lazy point-to-point planner* initialised with a *skeleton-heuristic* and *Euclidean assumption* for *House-W*. The *skeleton-heuristic* reduced planning times for *House-W* but due to the overestimation of local connectivity tour lengths were longer than tours solved under the *Euclidean assumption*.

However, given that the intended robotic platform is not expected to move quickly through the tank environments, a significant increase in tour length will disproportionately increase the time it takes to perform the coverage task. In relation to online planning, for a heuristic to decrease coverage planning times to only increase the time it takes to execute the plan, it is not a suitable approach. While the LPP does not in any way solve plans to optimality, it should strive to provide a feasible solution that will reduce planning times and overall tour lengths.

The resultant tours of *2x2m* and *House-W* in Figures 6-6 and 6-7 provide a visual representation of where the inflated tour lengths come from. A visual comparison between tour segments within obstacle free areas, show that there are noticeably longer routes through these areas compared to the solutions solved with a *Euclidean assumption*. The cause of these longer routes was the *skeleton-heuristic* overestimating connectivity within these local areas.

Simplified examples, demonstrating the overestimation in local areas for non-obstacle and obstacle-filled environments, are illustrated in Figure 6-8. Configurations that are adjacent are not guaranteed to be appropriately approximated using the *skeleton-heuristic*. As the skeleton will converge to the topological and geometrical centre of the environment, there is no guarantee that the branches of adjacent configurations will connect together in subsequent iterations to form the shortest path between them (Figure 6-8a).

Skeleton branches are determined by;

1) the order and direction in which the thinning is applied, and/or
2) the rotation of the environment (Cornea, 2007; Wheare, 2018).

The additional rule to the Lee's thinning algorithm (Section 6.5.1) does not attempt to alter the direction of these branches to ensure local connectivity is connected correctly. It merely serves as a process to connect the configuration to the skeleton by creating their own branches in the thinning process. As a result, adjacent configurations can appear distant and branches in the thinning process. As a result, adjacent configurations can appear distant and distant configurations may appear closer (Figure 6-8b). Figure 6-9 provides some examples of adjacent configurations having their distances overestimated due to the branches created by the thinning process for *2x2m* and *House-W*.

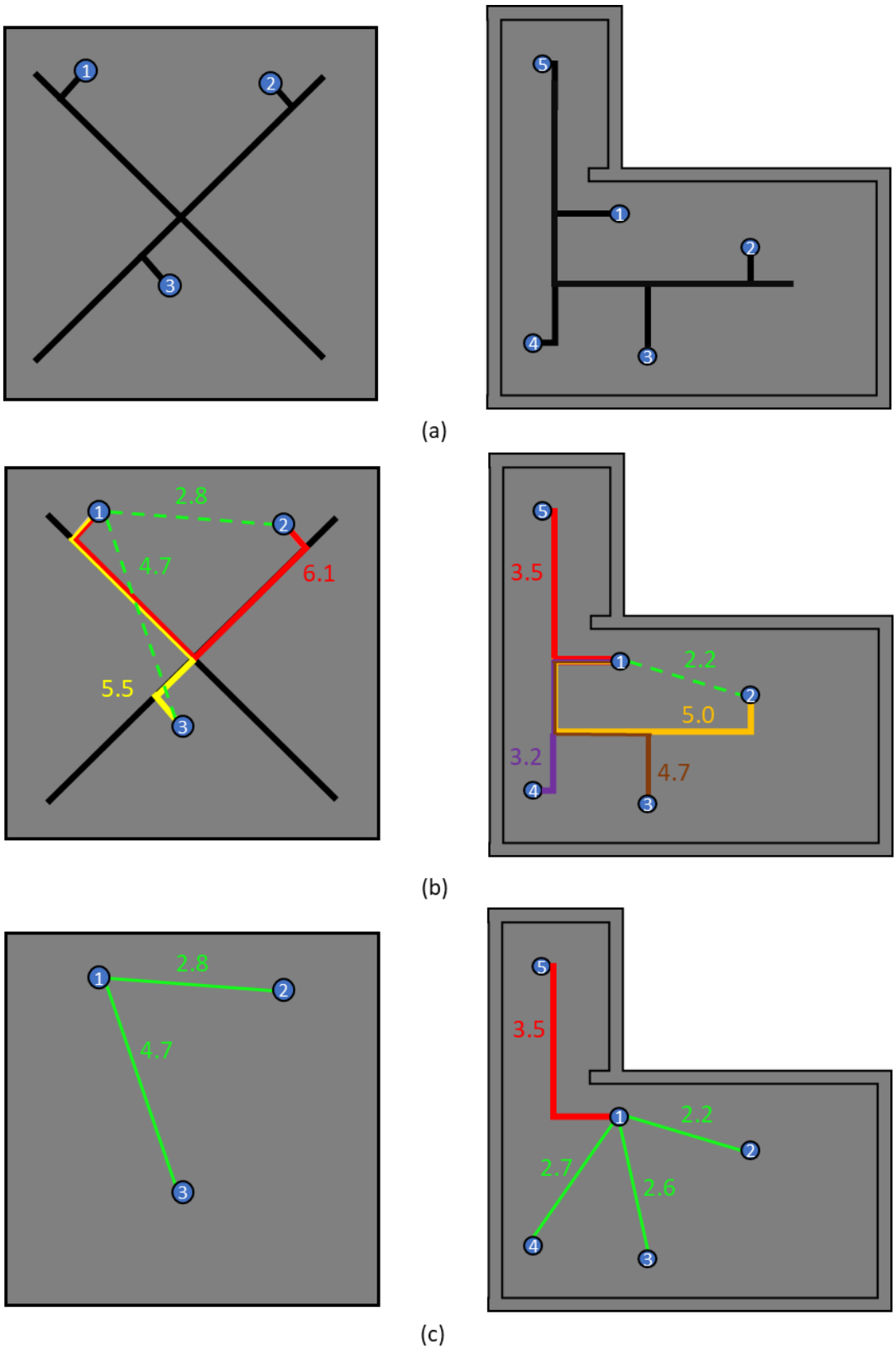


Figure 6-8: Example scenarios of the overestimation in non-obstacle filled and obstacle filled environments. (a) The representative skeletons of each environment with attached configuration branches. (b) The estimated distances recorded by the skeleton can overestimate local connectivity between configurations. (c) A line-of-sight check between configurations restores the *Euclidean assumption* for configurations not separated by an obstacle. This creates a hybrid distance metric that better approximates the connectivity between the configurations within various types of environments.

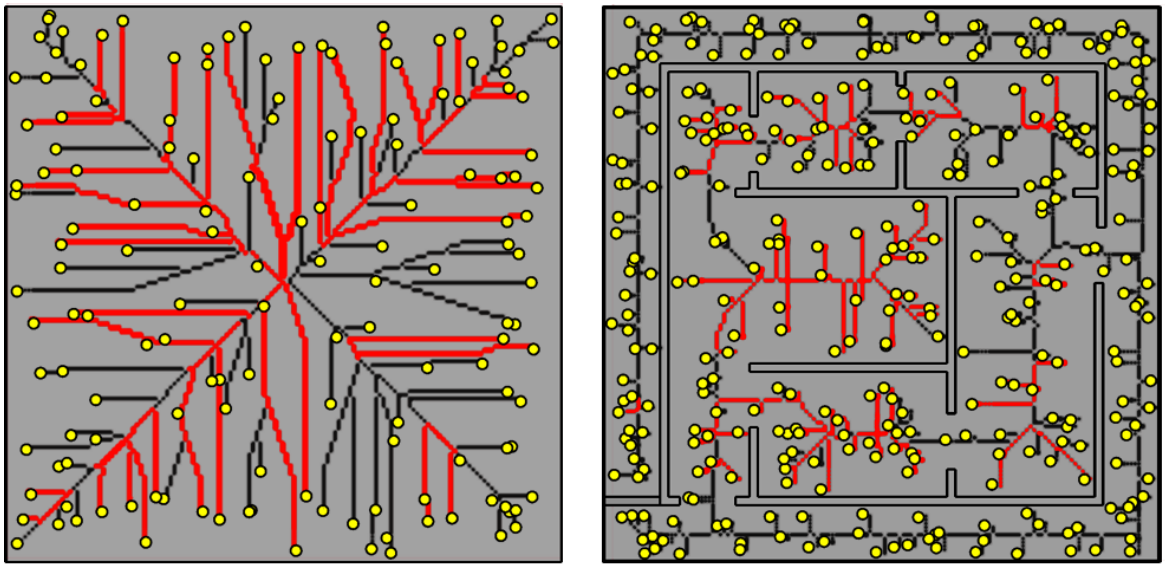
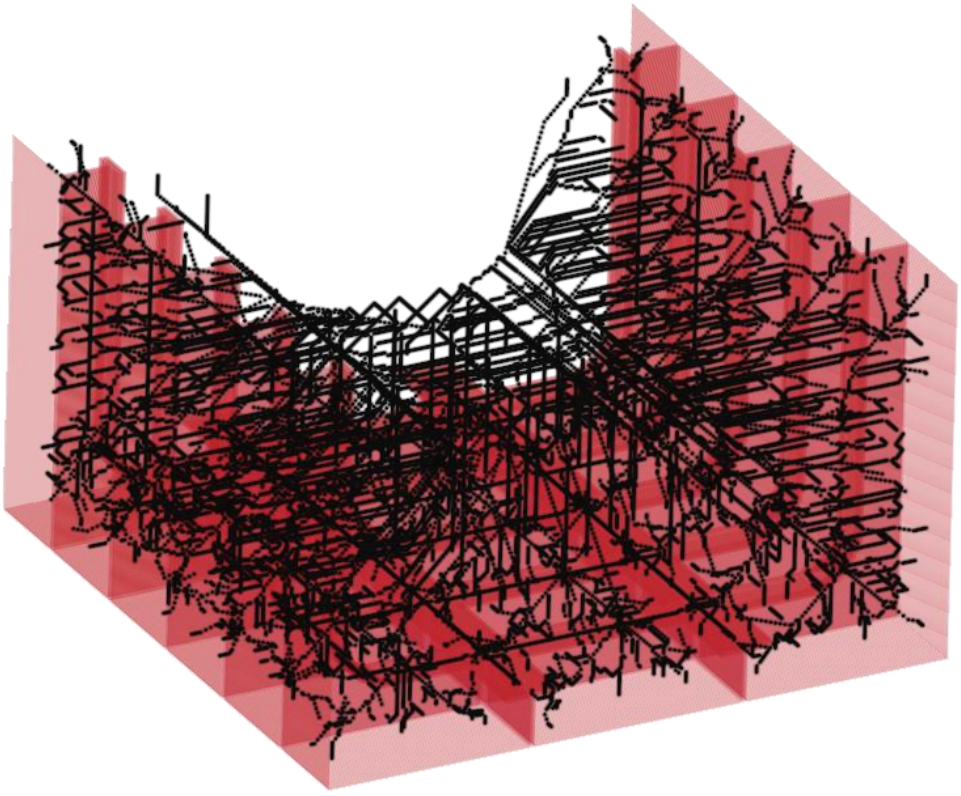


Figure 6-9: Examples of skeleton branches overestimating connectivity between adjacent configurations (red) for (a) *2x2m* and (b) *House-W*.

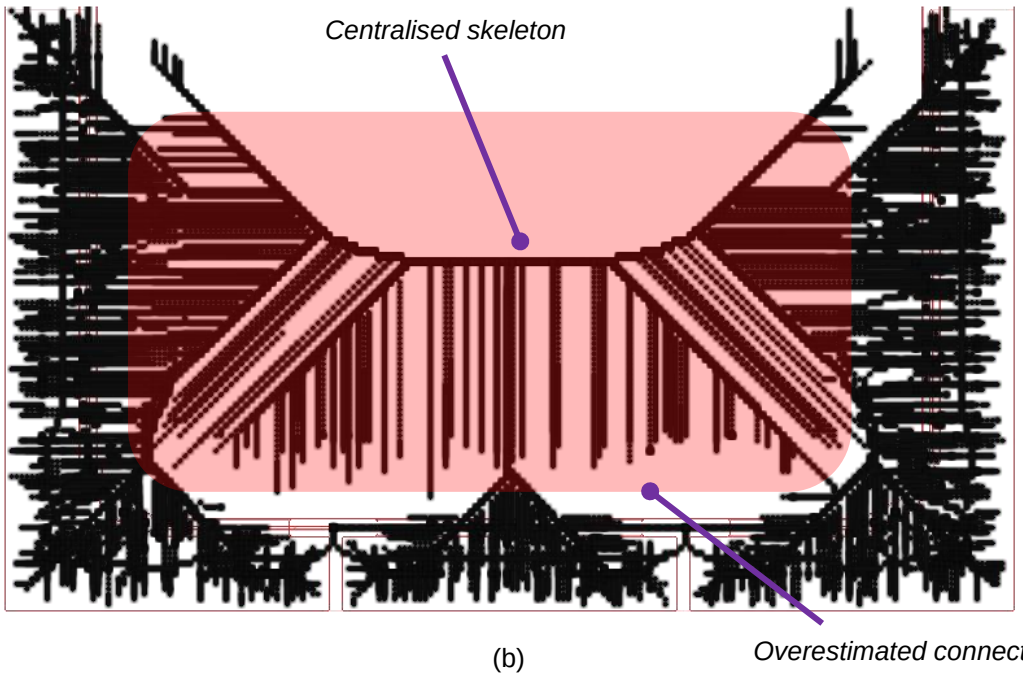
The overestimation within local areas gives a false representation of the connectivity and consequently has a negative impact on the TSP solver attempting to formulate a solution. As all the paths are overestimated, upon evaluating this first set of paths, all the evaluated paths become shorter than the skeleton distance. From the perspective of the TSP solver, in the next iteration, the planning problem presents a tour that is significantly shorter than any other alternative and is likely to be taken as the best tour available. The *skeleton-heuristic*, within local areas, has the reverse effect to what the Euclidean does, as the shorter paths become highly prioritised for the next iteration. Excluding the minor variability that is still present in TSP solutions, all planning problems evaluate approximately the same number of paths as configurations in the problem (Table 6-1).

The results suggest that the more open space an environment contains, the worse the degradation of tour quality will become. *Tank* and *Tank-P4* reported an 84% to 86% increase in tour lengths over the Euclidean. Figure 6-10 shows the resultant skeleton for *Tank*. As all the branches of the skeleton converged to the centre, more configurations within the open space were overestimated.

This analysis concludes that using an unrefined skeleton to inform the LPP about the connectivity between configurations is not appropriate due to overestimation between local connectivity. The purpose of the skeleton was to place a lower priority on paths that are further away. It was expected that the *skeleton-heuristic* may overestimate paths.



(a)



(b)

Figure 6-10: The open space of *Tank* creates overestimation of the skeleton. (a) The resultant skeleton of *Tank*. (b) The centeredness of the skeleton in open space promotes overestimation due to the skeleton branches (highlighted in red).

Overestimating paths that were further away was acceptable as the focus of this heuristic was to reduce the likelihood of these paths being evaluated in the TSP solution. However, since tours are generally comprised of paths between local configurations, overestimating nearby connections means overestimating the metric most used by the LPP, severely degrading effect on local tour quality.

Providing a better estimation on the connectivity creates a TSP problem that is dominated by local connectivity, therefore reducing the number of path evaluations required to solve the planning problem. The results show that the *skeleton-heuristic* is capable of achieving all these goals but at the expense of tour quality due to the overestimation in local connectivity. Consequently, the effectiveness of the *skeleton-heuristic* does not meet all the criteria as a suitable heuristic to replace the Euclidean as it degrades the tour quality of even the simplest planning problems.

6.7 The Hybrid-heuristic

A suitable approach to correct the overestimation caused by the skeleton branches was found by reinstating the Euclidean distance metric in local areas. After the skeleton was generated, a line-of-sight check was made between all configurations. Configurations that did not have line-of-sight retained the skeleton distance as an estimate (Figure 6-8c) while those configurations that did have line-of-sight had the Euclidean distance reinstated. Combining both distance metrics into the same adjacency matrix created a *hybrid-heuristic* that restored the lower bound of the LPP to be Euclidean (Equation 6-3) and remove the overestimation that was generated by the skeleton in local areas.

$$Euclidean \leq Actual\ length\ \leq Skeleton \tag{6-3}$$

To determine if two configurations should be connected via a direct path, a ray was cast between all configurations. If the ray cast between two configurations was free of collisions, the skeleton distance is replaced with the lower bound Euclidean distance. This extra line-of-sight check serves as a cheap approximation to solving an exact motion plan. If a motion planner was called for a pair of assumed locally connected configurations, the result of the path will not likely exceed the initial Euclidean distance.

Planning environments similar to *2x2m* and *6x6m* will receive no benefit from employing this *hybrid-heuristic* as it will create an adjacency matrix identical to the *Euclidean assumption*. Solving the MPP over these environments with the *hybrid-heuristic* only will

increase planning times due to the overhead calculating of the skeleton. It was expected that the effectiveness of this *hybrid-heuristic* will increase somewhat proportionally to the quantity and complexity of obstacles and obstructions within the environment.

6.8 Coverage Planning with the Hybrid-heuristic

The benchmark experiment was repeated to determine the effectiveness of the *hybrid-heuristic* to improve upon the tour length quality and whether it could solve coverage plans in similar times to the *skeleton-heuristic*. A relative pairwise comparison was conducted between the trials of this experiment against the results of the benchmark experiment that used the *additional termination conditions* (Section 5.6). A parametric Paired Samples T-Test ($\alpha = 0.01$) was conducted to complement the relative performance and determine if a statistically (p) and practically significant result (d) is present between the trials.

6.8.1 Computational Observations and Results

Tables 6-5 and 6-6 present the planning data for each of the trials and Tables 6-7 and 6-8 contain the relative performance and statistical and practical comparisons between trials. An analysis of results indicated the *hybrid-heuristic* had a positive impact on planning times for obstacle-filled environments and a negative impact for non obstacle-filled environments.

As expected, non obstacle-filled environments *2x2m* and *6x6m* defaulted to a Euclidean-based solution when using the *hybrid-heuristic*. All trials experienced longer planning times ($p < 0.001$, Tables 6-6 and 6-7) due to the time it took to generate the *hybrid-heuristic*. All tours were solved with similar path lengths as the benchmark experiment ($p = 0.19$), taking on average the same number of planning iterations ($p = 0.31$) and path evaluations to solve the planning problem ($p = 0.10$). The minor variation present in the tour lengths, which are under 1%, can be attributed to the variability of the TSP solver and the *additional termination conditions* (Sections 5.4 and 0).

Obstacle-filled environments, *House*, *House-W*, *Tank* and *Tank-P4* all reported a positive result to using the *hybrid-heuristic*. The number of path evaluations and planning iterations were significantly reduced across all problems ($p < 0.001$). Path evaluations were reduced by 51.8% and 40.9% for *Tank* and *Tank-P4* respectively. *House* recorded a 30% reduction with *House-W*, as seen in the skeleton experiment (Section 6.6), reducing by 99.8%. As fewer paths were being evaluated, the LPP converged faster. Consequently, planning iterations were reduced by 78.5% to 99.8% across these environments.

Table 6-5: Planning times, tour lengths and *lazy point-to-point planner* attributes for all planning scenarios using the *hybrid-heuristic*.

Model	Overall Time (s)		MPP Time (s)*		Tour Length (m)		Iterations		Path Evaluations	
	$\bar{x}$	SD	$\bar{x}$	SD	$\bar{x}$	SD	$\bar{x}$	SD	$\bar{x}$	SD
2x2m	5.22	0.11	2.71	0.10	21.47	0.27	3.03	1.33	154.25	6.49
6x6m	46.71	5.39	24.75	5.51	159.42	1.06	32.76	10.68	1,368.24	68.10
House	5.29	0.38	2.61	0.38	33.94	0.56	5.09	1.86	337.32	15.75
House-W	12.52	3.66	9.79	3.66	35.70	0.47	7.90	3.63	356.27	20.33
Tank	138.76	11.69	106.03	11.67	148.94	1.06	30.38	9.53	1,718.84	73.17
Tank-P4	201.21	12.68	134.91	12.69	175.08	1.30	47.04	10.25	2,347.25	68.26

*Includes the time to create the initial estimate and solve the MPP MPP - Multi-goal Planning Problem

Table 6-6: Analysis of the time taken to create the *hybrid-heuristic* and the resultant time to solve the *multi-goal planning problem* (MPP).

Model	MPP Time (s)	Hybrid-heuristic*			Solving the MPP		
		$\bar{x}$ (s)	SD (s)	TT %	$\bar{x}$ (s)	SD (s)	TT %
2x2m	2.71	2.49	0.03	92.25	0.21	0.10	7.75
6x6m	24.75	7.16	0.08	30.88	17.57	5.51	69.12
House	2.61	1.59	0.04	61.88	1.02	0.37	38.12
House-W	9.79	1.58	0.04	18.56	8.21	3.66	81.44
Tank	106.03	63.56	0.36	60.71	42.46	11.69	39.29
Tank-P4	134.91	66.59	0.39	49.80	68.31	12.70	50.20

*Includes voxelization, thinning and path finding MPP - Multi-goal Planning Problem
TT - Time Taken as a percentage of MPP time

Table 6-7: Relative performance and percentage difference between trials using the *Euclidean assumption* and the *hybrid-heuristic.*

Model	*Overall Time*		*MPP Time*		*Tour Length*		*Iterations*		*Path Evaluations*	
	RP	*PD %*	*RP*	*PD %*	*RP*	*PD %*	*RP*	*PD %*	*RP*	*PD %*
2x2m	*1.90*	*89.74*	*16.15*	*1515.41*	*0.99998*	*-0.002*	*1.16*	*16.19*	*0.99*	*-1.20*
6x6m	*1.24*	*23.86*	*1.81*	*81.21*	*0.99999*	*-0.001*	*1.24*	*24.00*	*1.00*	*0.44*
House	*0.56*	*-43.78*	*0.43*	*-56.90*	*1.00003*	*0.003*	*0.21*	*-78.52*	*0.61*	*-38.58*
House-W	*0.002*	*-99.82*	*0.004*	*-99.86*	*1.00017*	*0.017*	*0.001*	*-99.93*	*0.03*	*-97.12*
Tank	*0.35*	*-64.90*	*0.29*	*-70.53*	*1.00004*	*0.004*	*0.10*	*-89.75*	*0.48*	*-51.85*
Tank-P4	*0.53*	*-46.82*	*0.44*	*-56.07*	*1.00021*	*0.021*	*0.19*	*-80.80*	*0.59*	*-40.95*

MPP - Multi-goal Planning Problem
RP - Average of relative performance between each trial.
PD - Average of percentage differences between each trial.

Table 6-8: Statistical and practical significance between trials using the *Euclidean assumption* and the *hybrid-heuristic.*

Model	*Overall Time*		*MPP Time*		*Tour Length*		*Iterations*		*Path Evaluations*	
	p	*d*	*p*	*d*	*p*	*d*	*p*	*d*	*p*	*d*
2x2m	*<0.001*	*23.49*	*<0.001*	*28.07*	*0.19*	*-*	*0.44*	*-*	*0.01*	*-*
6x6m	*<0.001*	*1.50*	*<0.001*	*1.49*	*0.40*	*-*	*0.31*	*-*	*0.10*	*-*
House	*<0.001*	*2.17*	*<0.001*	*2.19*	*0.78*	*-*	*<0.001*	*2.55*	*<0.001*	*5.02*
House-W	*<0.001*	*9.71*	*<0.001*	*9.71*	*0.06*	*-*	*<0.001*	*11.72*	*<0.001*	*11.37*
Tank	*<0.001*	*4.29*	*<0.001*	*4.30*	*0.77*	*-*	*<0.001*	*5.27*	*<0.001*	*9.79*
Tank-P4	*<0.001*	*3.25*	*<0.001*	*3.25*	*<0.001*	*0.029*	*<0.001*	*4.43*	*<0.001*	*9.01*

MPP - Multi-goal Planning Problem *p - Paired samples t-test ($\alpha < 0.01$)* *d - Cohen's d effect size*

As a result of evaluating fewer paths, the number of paths retained in the final solution for obstacle-filled environments increased. *Tank* and *Tank-P4* retained 70% to 75% of calculated paths respectively while *House* and *House-W* retained 91.3% to 95.4% (Figure 6-11; Table 6-9). Compared to the analysis performed over the trials solved using the *Euclidean assumption* (Figure 5-5; Table 5-11), fewer RRTs were being evaluated across all obstacle-filled environments. No planning problem recorded more than 3% of the total paths evaluated to be RRTs. As a result, upwards of 96.8% of path evaluations were solved as direct paths. The fact that so few RRT paths were being evaluated and a higher rate of paths evaluated were being retained strongly indicates that the *hybrid-heuristic* better informs the LPP about the connectivity of the environment of paths before planning.

Correcting the overestimation within local areas enabled the LPP to solve tour lengths for all obstacle-filled environments within 1% of the Euclidean trials. Besides *Tank-P4*, no statistical significance was found between the tour lengths ($p = 0.06$). While *Tank-P4* reported a statistical significance ($p < 0.001$), the 0.02% increase in tour length was not practically significant ($d = 0.029$). Again, the slight variation can be a result of the reported variability of TSP solver generating solutions for larger planning problems or the randomness of the RRT paths chosen in the final solution.

Given that fewer paths are being evaluated in the obstacle-filled environments planning problems, similar to the results of the *skeleton-heuristic*, it has resulted in faster MPP solution times ($p < 0.001$) and overall planning times compared to the Euclidean trials ($p < 0.001$). Overall planning times for *Tank* and *Tank-P4* improved by 64.9% and 46.8% respectively, and for *House* and *House-W* improvements of 43.8% and 99.8% were recorded respectively.

As the same covering sets were used, creating the *hybrid-heuristic* had similar computational times to the skeleton creation (Table 6-6). The computation of the *hybrid-heuristic's* line-of-sight checks were negligible on planning times. Therefore, the creation of the *hybrid-heuristic* had no additional impact on overall planning times when compared to creation times of the *skeleton-heuristic*.

The minor differences in planning times between using the *skeleton-heuristic* and the *hybrid-heuristic* was due to slightly longer MPP times. As the connectivity was better represented, it required slightly more path evaluations to solve the planning problem. While the results indicated that planning times increased by approximately 50% for *Tank* and *Tank-P4* and

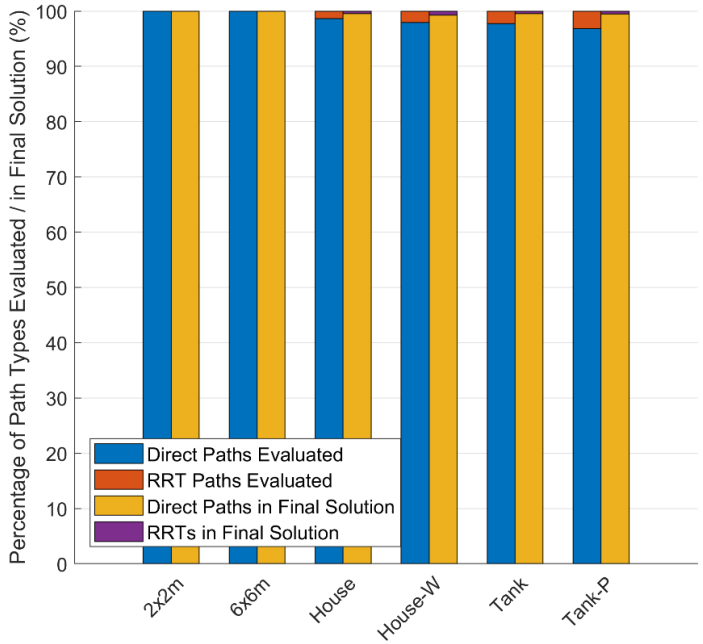


Figure 6-11: The introduction of the *hybrid-heuristic* increases the number of paths being retained in the final solution.

Table 6-9: Analysis of the number of path types evaluated, taken and retained in the final solution for trials solved under the *hybrid-heuristic*.

Model	*Average # of paths in final solution*	*Average # of paths evaluated*	*Evaluated Path Breakdown*				*Final Plan Path Breakdown*				*Calculated Path Retention*		
			Direct (Count / %)		*RRT (Count / %)*		*Direct (Count / %)*		*RRT (Count / %)*		*Direct (%)*	*RRT (%)*	*Combined (%)*
2x2m	*152.00*	*154.25*	*154.25*	*100.00*	*0.00*	*0.00*	*152.00*	*100.00*	*0.00*	*0.00*	*98.54*	*0.00*	*98.54*
6x6m	*1,011.05*	*1,368.24*	*1,368.24*	*100.00*	*0.00*	*0.00*	*1,011.05*	*100.00*	*0.00*	*0.00*	*73.89*	*0.00*	*73.89*
House	*321.80*	*337.32*	*332.53*	*98.62*	*4.79*	*1.38*	*320.25*	*99.52*	*1.55*	*0.48*	*96.31*	*32.36*	*95.40*
House-W	*325.35*	*356.27*	*348.75*	*97.92*	*7.52*	*2.08*	*322.88*	*99.24*	*2.47*	*0.76*	*92.58*	*32.85*	*91.32*
Tank	*1,301.85*	*1,718.84*	*1,679.62*	*97.72*	*39.22*	*2.28*	*1,295.58*	*99.52*	*6.27*	*0.48*	*77.14*	*15.99*	*75.74*
Tank-P4	*1,649.00*	*2,347.25*	*2,272.66*	*96.83*	*74.59*	*3.17*	*1,640.18*	*99.47*	*8.82*	*0.53*	*72.17*	*11.82*	*70.25*

upwards of 200% for *House*, they equated to no more than a few extra minutes to obtain a tour length of similar quality of the Euclidean trials. This is considered to be a suitable compromise that is acceptable in this case, given the significant reduction in the number of path evaluations, MPP times and overall planning times.

Finally, a notable beneficiary of the *hybrid-heuristic* is *House-W*. Originally, solving the MPP for *House-W* using a *Euclidean assumption* saw 93.5% of the 12,717 paths evaluations being resolved by RRTs (Figure 5-5; Table 5-11). Only 0.02% of these evaluated RRTs using the *Euclidean assumption* were used in the final tour. When solving the MPP for *House-W* using the *hybrid-heuristic*, 2% of the 356 paths evaluated were RRTs, with 32% of these being retained in the final tour (Figure 6-11; Table 6-9). For both the Euclidean and *hybrid-heuristic* estimations of the connectivity, *House-W* used approximately the same number of direct and RRTs paths in the final solution. Evaluating fewer RRTs has led to a 99.8% improvement in overall and MPP times, with less than 1% degradation in tour quality.

6.8.2 Discussion

The results of this experiment demonstrate that for the given planning problems, the *hybrid-heuristic* has successfully managed to reduce planning times by minimising the number of path evaluations required to solve a planning problem and produce quality solutions equivalent to the coverage plans solved under the *Euclidean assumption*. Introducing an additional line-of-sight check to restore the *Euclidean assumption* in local areas that were overestimated by the skeleton enables the *hybrid-heuristic* to satisfy all the criteria for a suitable replacement (Section 6.3).

Using a skeleton provided enough information about the environment such that the connectivity of the configurations could guide the LPP to converge to solutions faster than using the *Euclidean assumption*. The LPP became more efficient as fewer paths were being evaluated, resulting in more paths being retained in the final solution. As the connectivity was better represented, fewer RRTs were evaluated across all planning problems. The LPP evaluates fewer RRTs, which suggested that the TSP solver was not choosing the edges it would have under a *Euclidean assumption*. As a result, larger planning problems, *Tank* and *Tank-P4*, solved in under four minutes and *House-W*, which was previously the hardest planning problem, solved in 12.5 seconds (Table 6-5).

Despite substantial improvement in planning times for obstacle-filled environments, the

experiment demonstrated that using a skeleton to formulate an understanding of the environment is not applicable for all environment types. Non-obstacle filled environments, *2x2m* and *6x6m*, did not receive any computational improvement using the *hybrid-heuristic*. While planning times increased compared to the trials using the *skeleton-heuristic*, the *hybrid-heuristic* removed the overestimation in areas of local connectivity by reinstating the *Euclidean assumption* where applicable. This produces a better outcome compared to using the *skeleton-heuristic* that resulted in shorter planning times but significantly longer tour lengths. While the *hybrid-heuristic* may not be applicable for most simplistic environments, it has made fast planning times in complex environments possible.

6.9 Application of the Hybrid-heuristic over a Life-like Office Space

To demonstrate the applicability of using topological skeletons in complex spaces beyond the planning environments tested over the last three chapters, the *hybrid-heuristic* was tested over a life-like office floor plan. The 70x40x2m office floor, has several rooms, wide-open spaces and long corridors that will underestimate many paths, causing great difficulty for the LPP under the *Euclidean assumption* to stabilise quickly to a solution.

The robot constraints placed on this planning problem, applied a spherical 6-DOF holonomic robot platform to represent UAV platform to perform the inspection. A 300mm sphere was used as the collision model and the viewing constraints set to range a of 0.5-1.0m with a Field of View (FOV) of ±70°. The 70x40x2m office space was voxelised to a 70mm resolution. The *additional termination conditions* were applied to both LPPs. To witness the benefits of applying the *hybrid-heuristic* across a significantly large and complex environment, no time limit was applied to the planning problems.

The statistical analysis of the planning data acquired over the trials presented in Tables 6-10 to 6.12 show the positive impact the *hybrid-heuristic* has on the LPP. Figure 6-12 illustrates the resultant skeleton over office space and tour for the LPP using the *hybrid-heuristic*. Coverage plans that took on average 1.1 days to solve under a *Euclidean assumption* took approximately 8.8 minutes to solve using the *hybrid-heuristic*. Of the 8.8 minutes needed to solve the MPP with the *hybrid-heuristic*, 36% of the overall time was spent generating the skeleton. Due to time taken for the LPP under the *Euclidean assumption* to find a solution, only five trials were required to determine that the Euclidean LPP would not solve the planning problem faster than the *hybrid-heuristic*.

Table 6-10: Statistical analysis of the planning times between Euclidean and *hybrid-heuristic* trials.

Heuristic	*Trials*	*Configurations*	*Overall Time (s)*		*CSP Time (s)*		*MPP Time (s)*	
			$\bar{x}$	*SD*	$\bar{x}$	*SD*	$\bar{x}$	*SD*
Euclidean	*5*	*1429 - 1456*	$95.6x10^3$	$17.4x10^3$	*29.8*	*0.7*	$95.6x10^3$	$17.4x10^3$
Hybrid	*20*	*1411 - 1465*	*533.4*	*147.2*	*30.7*	*2.3*	*502.7^*	*147.8*

^MPP time includes calculating the hybrid-heuristic (Table 6-11)
CSP - Coverage Sampling Problem
MPP - Multi-goal Planning Problem

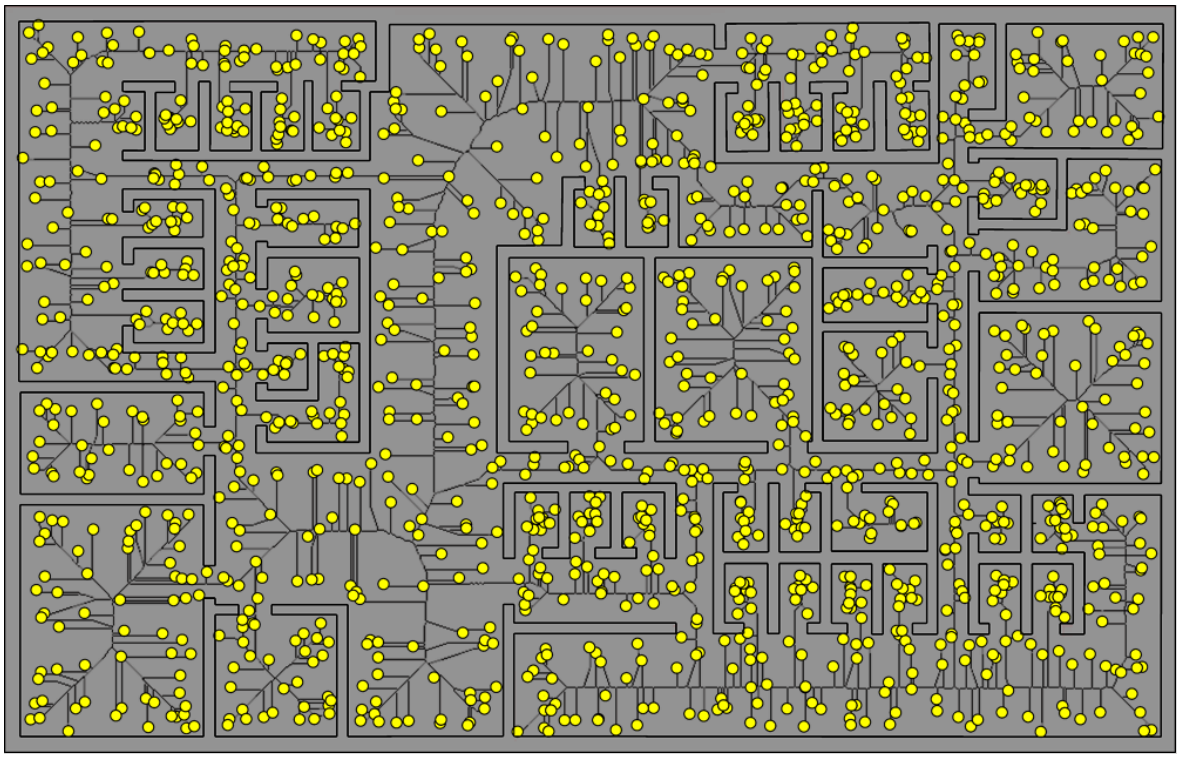
Table 6-11: Time taken to create the *hybrid-heuristic* and the resultant time to solve the *multi-goal planning problem* (MPP).

Heuristic	*MPP Time (s)*	*Hybrid-heuristic**			*Solving the MPP*		
		$\bar{x}$ *(s)*	*SD (s)*	*TT (%)*	$\bar{x}$ *(s)*	*SD (s)*	*TT (%)*
Hybrid	*502.7*	*168.9*	*1.9*	*36.4*	*333.8*	*147.6*	*63.6*

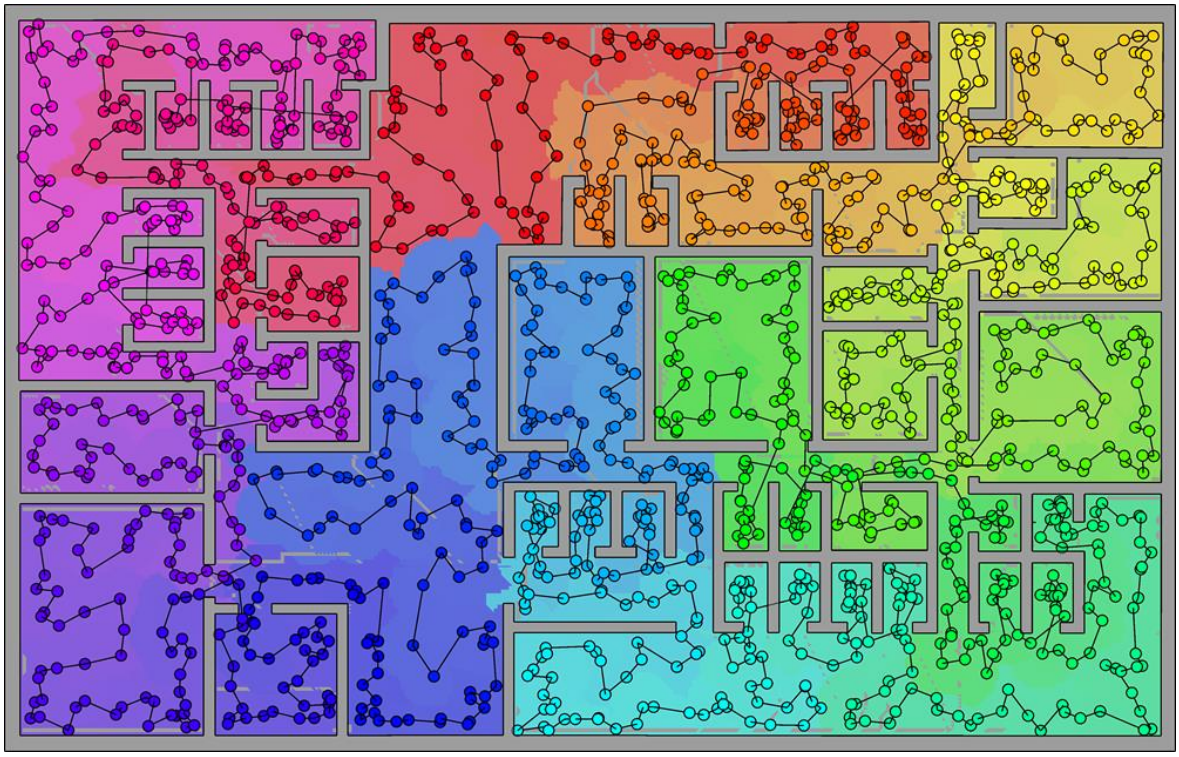
**Includes voxelization, thinning and path finding*
MPP - Multi-goal Planning Problem
TT - Time Taken as a percentage of MPP time

Table 6-12: The resultant planning attributes of the *lazy point-to-point planner*.

Heuristic	*Tour Length (km)*		*Planning Iterations*		*Path Evaluations*	
	$\bar{x}$	*SD*	$\bar{x}$	*SD*	$\bar{x}$	*SD*
Euclidean	*1.71*	*0.1*	$28.1x10^3$	$6.7x10^3$	$15.2x10^3$	$20.7x10^3$
Hybrid	*1.72*	*0.1*	*278.6*	*124.4*	*3051.6*	*354.6*



(a)



(b)

Figure 6-12: Planning with the *hybrid-heuristic* over a large complex environment. (a) The resultant skeleton of the office floor space with connected configurations. (b) The final tour. Grey primitives on the floor of the office in (b) are due to anomalies in the mesh model that made these primitives unobservable.

As expected, solving the LPP using the *hybrid-heuristic* has resulted in a significant reduction in the number of path evaluations and planning iterations ($p < 0.001$, $d > 0.8$). Overall, between the LPPs, the *hybrid-heuristic* plans only saw a 0.6% increase in tour length. As this is under 1%, the variation could be due to either variation of the LPP or the variation of the RRT paths, which would be expected to be significant for the same pair of configurations. However, considering the improvement in overall planning times, the tour degradation is negligible.

An interesting finding of this experiment is that neither coverage planner achieved 100% coverage of this environment but 92%. Full coverage could not be achieved due to anomalies with 3D mesh model that was generated from a modified 2D floorplan. However, given the CSP times are equivalent, strongly indicates that the *primitive rejection count* (Section 4.2.3) worked as expected. The CSP did not continue to sample primitives beyond the *primitive rejection count* and subsequently these primitives were recorded as *unobservable*. The primitives that could not be observed are highlighted in grey in Figure 6-12b.

The importance of these results show that the *offline sampling-based coverage planner* is capable of solving very large and complex planning problems, providing the LPP is supplied with a sufficient estimation of the connectivity connecting configurations. The introduction of the *hybrid-heuristic* along with the *additional termination conditions* has aided the *offline sampling-based coverage planner* to overcome one of the main influences that contribute to long computational times, the environment. These experiments provide evidence that the improvements developed thus far in this thesis address the two main concerns that were identified that would impact *adaptive coverage path planner* online sufficiently reporting *unobservable primitives* (Section 3.4.3) and long computational times (Section 3.4.4). As these improvements have been proven to work offline, these algorithms can be deployed online with minimal modification for online use. However, consideration has to be given when using the *hybrid-heuristic* as it is not applicable to all planning scenarios.

6.10 Applicability of the Offline Planning Results to Online Planning Proposals to Solve the STIPP

Given the results presented in this chapter, the *full replan strategy* still presents as a safe and reliable approach to the online replanning problem. The *hybrid-heuristic* will aid the LPP in handling any geometrical changes that would significantly impact planning times and

solutions solved under a *Euclidean assumption*. However, despite the benefits that the *hybrid-heuristic* would provide, there are concerns with implementing a *full replan strategy* to adapt the *offline coverage planner* to solve online planning problems.

Planning times for the representative tank structures are still within a few minutes. Again, this is due to the cost of solving the MPP. While the *hybrid-heuristic* reduces planning times by up to 47-60% and the number of path evaluations by 40-50% compared to using the *Euclidean assumption*, a complete replan will still incur a significant cost upon each replanning iteration. Even if a *full replan* was performed under the simplest planning constraints, the number of path evaluations required to solve a planning problem would still incur a significant cost, despite the possibility of retaining 75% of the path evaluated. Given that upwards of 2000 configurations are required to cover the representative tank environments, this is a consequence that has to be accepted given;

1) the type of coverage planner used to solve the planning problem,
2) the robotic platform constraints placed on the planning problem, and
3) the complexity of the environment for which the coverage planner is replanning.

Given the efforts over the previous chapters to minimise the computation of the LPP by reducing the number of planning iterations and path evaluations required to solve a planning problem, a *full replan strategy* only seeks to counter these efforts. Given the large number of path evaluations that are required to solve the representative tank environments, discarding the majority of the tour at each replanning iteration only increases the recalculation of the motion plans by a high-fidelity motion planner, whether it is performed during or after the solution is generated.

A full replan may be an appropriate strategy to the replanning problem if;

1) the planning problem (environment and covering set) is small in size,
2) minimal replanning updates occur during execution,
3) the changes significantly invalidate majority of the existing tour, or
4) high-fidelity motion planning is computationally inexpensive.

However, given the target environments are expected to be large in size, rules out the likelihood of the *full replan strategy* being the best option to solve the *submarine tank inspection planning problem* (STIPP).

As discussed in Section 3.5, a suitable online coverage planner needs to minimise the

computation of calculating a plan as well as minimising the computation of executing a coverage plan over the entire Inspection Planning Framework (IPF). Therefore, given the results of the investigations undertaken over the last three chapters into intrinsic behaviour of the *offline sampling-based coverage planner*, the *path repair strategy* presents itself as the better option to implement to solve STIPP. The results have shown that to make the sampling and path planning procedures efficient processes online, the planning problem supplied to these processes need to have constraints on its size and complexity. The improvements to the LPP alone have shown that the LPP can solve smaller planning problems quite quickly.

The *path repair strategy* proposes to confine the influence of the changes detected within the environments to highly localised regions, reducing the computational impact of producing a new tour upon each replanning update. Similar to the *hybrid-heuristic*, the *path repair strategy* will only be a viable solution if it can;

1) reduce overall planning times enough to compensate for calculating the impact of the changes, segmenting the current tour, replanning the changes and merging the resultant sub-plans to create a new plan, while
2) maintaining an acceptable level of tour quality degradation, as the final solution is not being solved globally.

If these conditions can be met, both the sampling and path planning procedures receive the benefit of working over smaller planning problems and because only what is needed to be replanned is being replanned, it will result in fewer paths required to be solved by both the LPP and the high-fidelity motion planner. In the next chapter, the *adaptive sampling-based coverage planner* using a *path repair strategy* is discussed in further detail.

6.11 Chapter Summary

In this chapter, the LPP was investigated further to determine the cause behind why so many path evaluations had to occur to solve a given planning problem. The initial *Euclidean assumption* underestimated the connectivity between configurations because it did not factor in the environment. This simplistic assumption prevented the LPP to solve efficiently over complex environments.

It was proposed to use the properties of topological curve-skeletons to form an adjacency matrix that factored in the influence of the environment on connectivity. A rule amendment

to Lee's *3D parallel thinning algorithm* enabled voxels containing the position of configurations to be connected to the skeleton during the thinning process. The results of the *skeleton-heuristic* showed that a significant reduction in path evaluations which led to shorter planning times compared to the Euclidean trials. However, overestimation of distances in local areas of connectivity by the skeleton significantly degraded tour quality.

An amended *skeleton-heuristic* that reinstated the Euclidean distance as the minimum lower bound in areas of locally connected configurations formed the *hybrid-heuristic* that mitigated the local overestimation. As a result, planning times with the *hybrid-heuristic* were comparable to the skeleton, with the tour quality within 1% of coverage plans being solved under the *Euclidean assumption*. The more informed heuristic achieved the goal of reducing planning times by reducing the number of path evaluations needed to solve a planning problem. While the *hybrid-heuristic* was not suitable for use in simple environments, an additional experiment of an office space, more closely representing a real-world environment, demonstrated the benefits topological skeletons had to offer for solving significantly large and complex offline coverage path planning problems.

Despite the significant reduction in path evaluations and planning times, it was decided that a *full replan strategy* would not be a suitable strategy to implement for an efficient online adaptive planner. The following chapter explores the second online adaptive proposal that implements the *path repair strategy*. This strategy bounds new features within *regions of interest* to reduce the computational effort required by the CSP and MPP processes to produce a solution.

Chapter 7
Adaptive Sampling-Based Coverage Planning using a Plan Repair Strategy

7.1 Introduction

In *Chapter 3*, an *adaptive sampling-based coverage planner* using a *plan repair strategy* was proposed as an alternative to the *full replan strategy* to solve the online replanning problem. The offline experiments documented in *Chapters 4 to 6* have shown that, providing there is no significant underestimation of the connectivity between configurations, planning can occur faster across smaller planning problems. Algorithmically, smaller planning problems will generate shorter solution times. Reducing the number of primitives and configurations required to be replanned generates a proportional reduction in the computation effort required by both the offline *coverage sampling problem* (CSP) and the *multi-goal planning problem* (MPP) algorithms.

In an effort to reduce the computational effort of preforming an online tour update, the *plan repair strategy* focusses exclusively on the *newly detected features* within the environment to determine the extent of the replanning effort. The *plan repair strategy* uses a *segment, replan and merge* approach to;

1) segment the current tour using a *region of interest* (ROI) to encapsulate changes within the environment to preserve any configurations and paths not affected by the changes,
2) replan new coverage and paths within the ROI to account for the new changes, and
3) merge, the new coverage (sub-plan) back into the preserved tour to create a new plan that accounts for the new features without having to replan an entirely new plan.

These three steps constitute the *adaptive sampling-based coverage planner* that uses a *plan repair strategy* to minimise the computational effort required by the offline sampling and path planning processes to perform an online update.

The separation allows the replanning effort to only occur within the ROI to reduce the number of existing viewing configurations that need to be replanned. Given the robot has a limited viewing capability, the inspection task under the current constraints (Section 3.6), will have upwards of 1500 viewing locations to cover a representative submarine tank environment (Section 4.3.3). Any effort to reduce the planning effort is essential to provide timely planning updates.

Contentions with the *plan repair strategy* is;

Question 1) *What is the computational impact of segmentation and merging of sub-plans on the planning times*?

Question 2) *What is the potential drop in tour optimality as the sub-plan approach does not solve the plan globally?*

The proposed *plan repair strategy* is only effective if the three processes of segmentation, replanning and merging can compute faster than the time required to replan the entire plan with minimal tour degradation. Similar to the criteria applied to the *hybrid-heuristic* (Section 6.3), it would be counterproductive for the *plan repair strategy* to decrease the computational cost of calculating a plan if it led to an overall increase in execution time. The experiments in the previous chapters suggest that a *full replan strategy* is likely to increase both the computational time of creating and executing an updated coverage plan. This chapter explores the adaptation of the *offline sampling-based coverage planner* into and *online adaptive sampling-based coverage planner* by introducing ROIs to aid in partially replanning new features into an existing plan. Figure 7-1 provides a visual representation of the *plan repair strategy* that was presented in Figure 3-7.

7.2 Transitioning from an Offline to Online Planning Problem

The transition from an offline to an online planning problem requires a reassessment of how the planning problem is presented to the coverage planner. New features within the environment have the potential to compromise the current plan. Therefore, it is the responsibility of the *online coverage planner* to ensure that all primitives, new and old, remain covered and the final plan stays connected a single collision-free tour. Since new

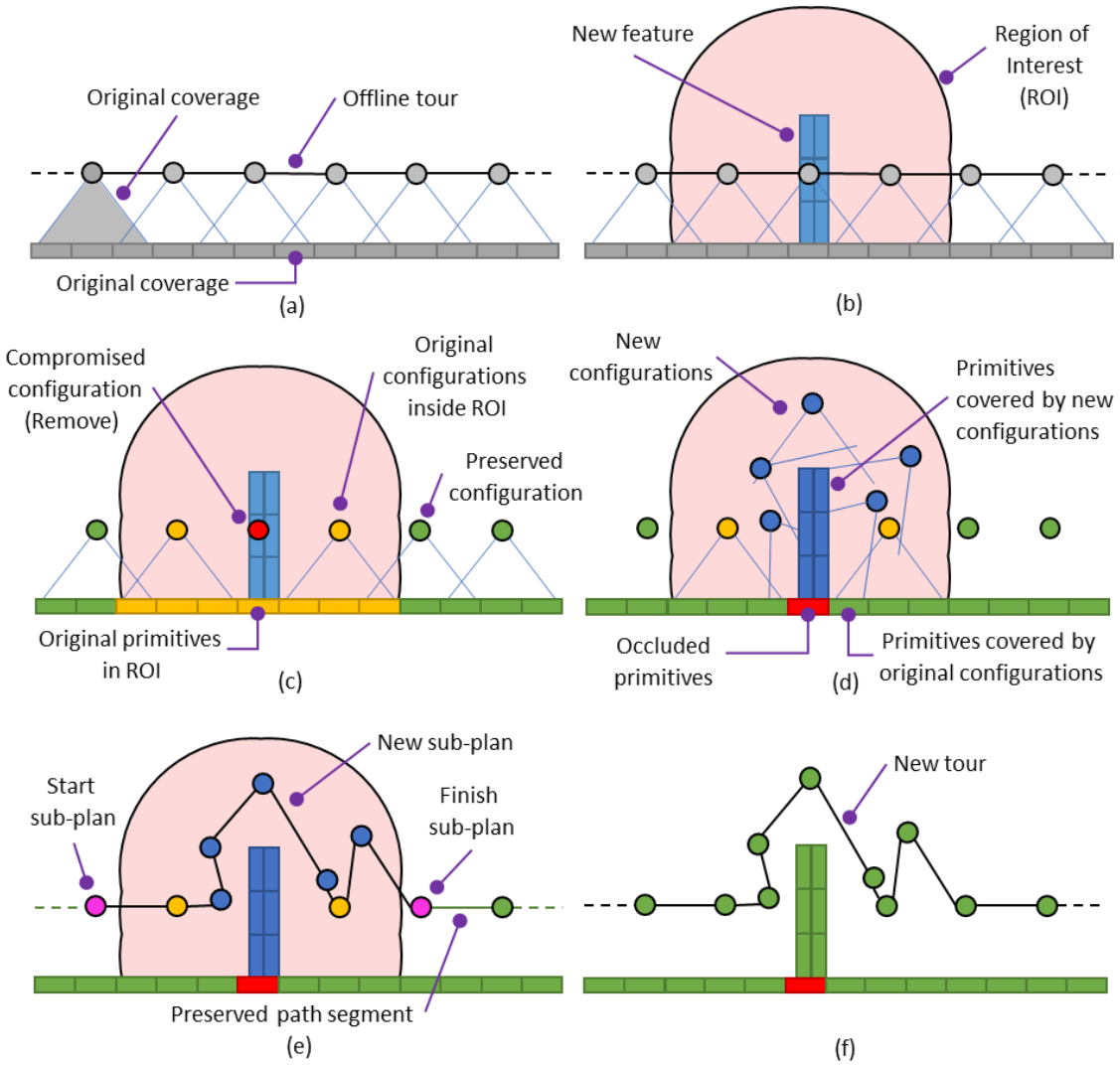


Figure 7-1: Conceptual diagrams of the *adaptive sampling-based coverage planner* implementing a *region of interest* (ROI) to bound the replanning effort to construct the plan repair strategy. (a) The offline plan over a known environment. (b) A ROI bounds the new primitives of the map update. (c) The original positions are validated against the ROI. (d) The *covering sampling problem* (CSP) and *set cover problem* (SCP) produce new configurations to cover all primitives in the ROI. (e) The MPP produces a new sub-plan that is connected back into the existing plan. (f) The plan is passed to the robot and will remain as the only plan until a new map update is provided.

primitives are expected to be introduced into the planning problem, which may invalidate a set of existing configurations, a new relationship between the primitives and configuration space is required to accurately represent the online planning problem.

This section discusses the transition to the online planning domain. It covers;

1) the introduction of a new *set system* (M,Q) that better represents the relationship between the primitive M and configuration Q spaces for an evolving environment,
2) the generation of a conceptual ROI within both the M and Q spaces to capture the appropriate replanning effort required to produce complete coverage plans, and
3) to address the impact new features will have on replanning problem.

7.2.1 Defining a New Set System Relationship

As discussed in *Chapter 3*, Englot and Hover (2012a) represented the relationship between the primitive space P and configuration space Q for an offline planning problem as a *set system* (P, Q). This *set system* maps configurations q_j generated in Q to a subset of primitives p_i in P. For the online planning problem, the introduction of new primitives into the planning problem invalidates *set system* (P, Q). Therefore, a new *set system* definition is required to accurately represent the online planning problem.

In the online planning problem, the mapping system supplies the adaptive coverage planner with a map update M. Under the assumption that the mapping system represents detected features as primitives, M will append the new features P^* over the original map P s.t. $M = \{P \cup P^*\}$, where P^* is the set of all new primitives p_k^* that comprise a set of newly identified structures s.t. $p_k^* \in P^*$. With the introduction of P^*, the original *set system* (P, Q) no longer holds true as P strictly refers to the original environment and introducing new features may cause existing tour configurations to now observe P^* or not observe P. Therefore, a new *set system* (M, Q) is defined to represent planning in the online domain. As the assumption that the outer boundaries of the environment are always known throughout the inspection (Section 3.6.1; *Assumption 1*), Q in the online planning problem, will remain the same. Figure 7-2 illustrates the two *set systems* (P, Q) and (M, Q) for the offline and online planning problems respectively.

Using M to reflect the current status of the environment allows P to always represent the previous understanding of the environment. As the mapping system is assumed to always append new features into the environment (Section 3.6.1; *Assumption 3*), when the coverage planner has replanned the current tour by incorporating new coverage for P^*, M becomes P at the next planning iteration (Figure 7-6). Assigning M to P at the conclusion of a replanning iteration allows the planner to assume that the previous iteration is equivalent to an offline plan. This recursive assignment allows features detected in the previous iteration to be treated as if it had already existed in the environment. This allows the *adaptive sampling-based coverage planner* to perform the same replanning functions irrespective to any previous planning update.

As the *set system* relationship between a configuration and the primitives it observes is necessary to maintain when replanning online, an additional term v is introduced to represent $p_i \subset P$ that is observed by q_j. A secondary term v^* is used as the recalculated

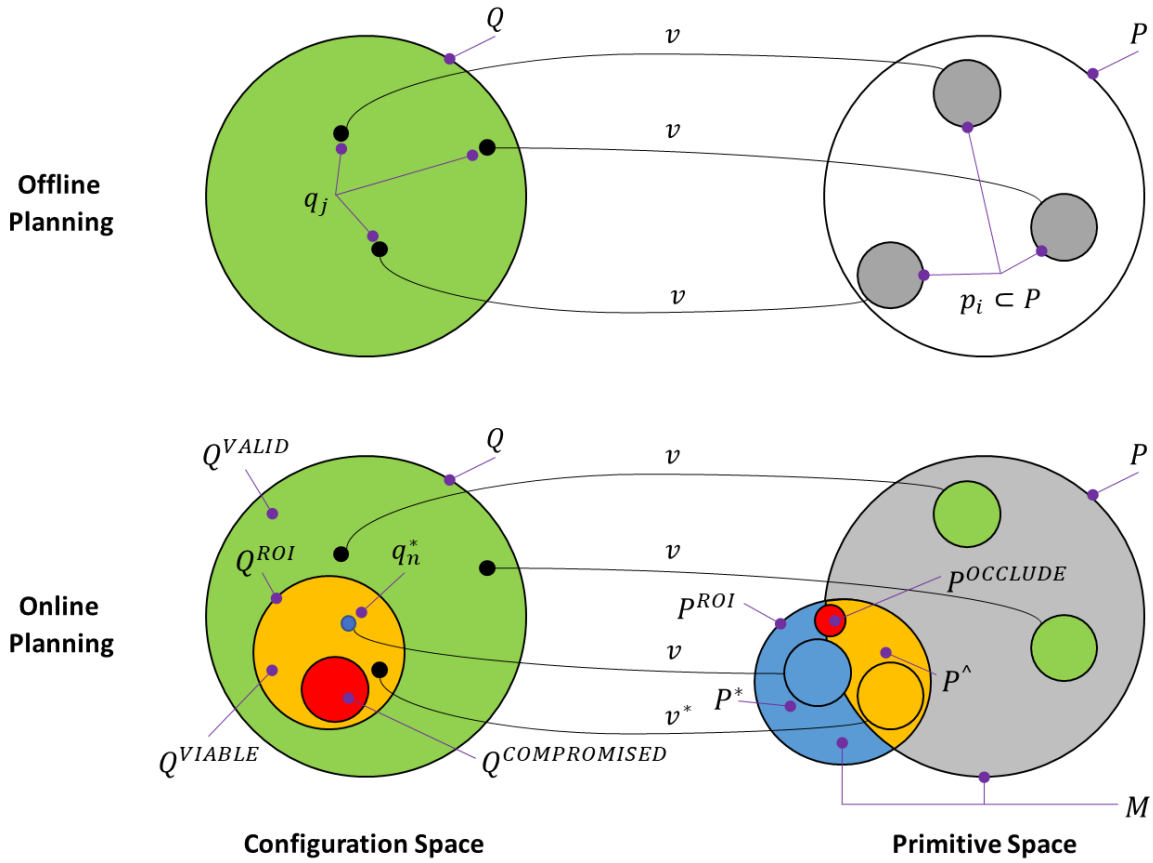


Figure 7-2: An updated set system relationship between the configuration space $\boldsymbol{Q}$ and the primitive space $\boldsymbol{M}$ for the online system. Each set is represented by a colour. The colours used for each set will be used to represent the same set in subsequent figures throughout this chapter.

coverage of q_j. The importance of v^* is discussed in Section 7.4 when validating existing configurations against the influence of P^*.

7.2.2 Defining a ROI within the Configuration and Primitive Spaces

A ROI, as depicted in Figure 7-2, will encapsulate the influence P^* has on P and Q, regardless of the geometric size of P^*. Within M, the ROI defines a space P^{ROI} that will always include P^* along with possibility of containing primitives of the existing structure. Consequently, within Q, the ROI defines a configuration space Q^{ROI} that is used as a metric to segment the current tour T[4]. Any $q_j \in Q^{ROI}$ become candidates for replanning and $q_j \notin Q^{ROI}$ are preserved and remain in the tour. The configuration space Q^{ROI} also defines a space in Q that is used to generate new configurations q_n^* that observe P^{ROI}. With P^* being fully encapsulated by the ROI, $p_k^* \in P^*$ can only be observed by configurations generated

[4] For consistency with the online planning definitions, T replaces G which was defined in Section 3.3.3 as the ordered set of configurations to creates a collision-free tour through Q.

within Q^{ROI}.

A ROI is created by bounding every $p_k^* \in P^*$ within its own ROI (ROI_k). Each ROI_k is calculated based on the camera's maximum field of depth (FOD_{max}) from the centre of each p_k^*. In the same way, FOD_{max} is used to create a subset of Q to generate samples to view P, using FOD_{max} as the metric to segment T, ensures that only $q_j \in T$ that can potentially be affected by P^* are subjected to replanning. Equation 7.1 defines Q^{ROI} as a subset of Q that is used to generate a set of configurations that are within the ROI. The function f_{xyz} projects elements of a set into the Cartesian space.

$$Q^{ROI} \triangleq \{ Q | \, dist(f_{xyz}(Q) - f_{xyz}(p_i^*)) < FOD_{max} \in \mathbb{R}, \forall p_k^* \in P^* \} \tag{7.1}$$

Figure 7-3 illustrates the geometrical representation of a ROI based on Equation 7.1. Figure 7-3a shows a ROI_k being generated around a single primitive based on FOD_{max} and Figure 7-3b illustrates the collective ROI formed though a cluster of individual ROI_k.

When a map update is supplied, it would be expected that the majority of the new primitives will be grouped around the new structures. However, by bounding each p_k^* within ROI_k allows for geometrically disconnected clusters of $p_k^* \in P^*$ to be treated as separate regions. By generating an individual ROI for each primitive, the planner has the ability segment T without having to make any assumptions about the environment. Replanning can occur throughout the environment simultaneously with no discrimination to size, shape or location.

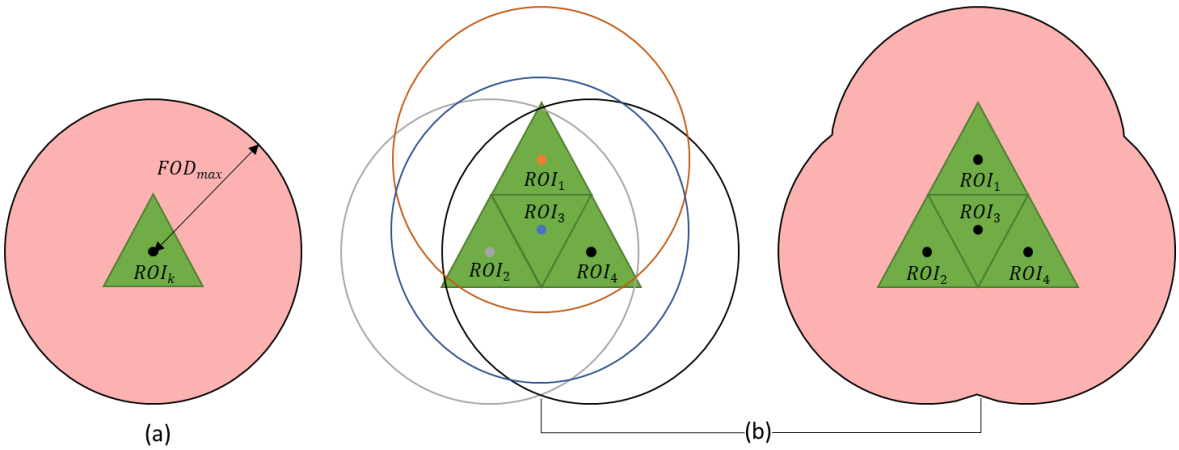


Figure 7-3: Generating the *region of interest* (ROI). (a) A single primitive produces its own ROI (ROI_k) based on the maximum field of depth (FOD_{max}). b) The collective of multiple primitives creates the conceptual ROI that is used limit the replanning effort over the environment.

Performing an evaluation based on a scale metric allows the solution to be solved generically across the space. It is a solution that can be applied to any situation and does not require parameterisation of an object or a complete object to perform, making it a suitable metric to replan over evolving features within the environment. As the ROI makes no assumptions about the environment, this approach adheres to the generic nature of the *offline sampling-based coverage planner*.

The ROI also serves to capture the influence P^* has on P. The introduction of P^* may invalidate, via collision or occlusion, the existing coverage v of q_j. If q_j has to be either removed or has some portion of its coverage occluded, the primitives it once observed may not be observable to other $q_j \in Q^{ROI}$. In the case q_j is removed from the coverage plan, the unique observations made by q_j will definitely be unobserved by the remaining $q_j \in Q^{ROI}$.

To guarantee the probabilistic completeness of the final coverage plan, discussed in further detail in Section 7.8, any configurations whose v is compromised due to P^* resulting in unobserved primitives, $P^{\#}$, must be included in P^{ROI}. Therefore, it is necessary to ensure that primitives that may lose their coverage ($P^{\#}$) are included into P^{ROI}.

To ensure all these primitives are captured within P^{ROI}, the coverage v of all $q_j \in Q^{ROI}$, $P^{\wedge}$ (Equation 7.2), are also added to P^{ROI} (Equation 7.3). By all adding all v of $q_j \in Q^{ROI}$ into P^{ROI} ensures that if configurations are removed, $P^{\#}$ will always be contained in P^{ROI}.

$$P^{\wedge} \triangleq \{v \mid \forall q_j \in Q^{ROI}\} \tag{7.2}$$

$$P^{ROI} \triangleq \{P^* \cup P^{\wedge}\} \tag{7.3}$$

Since Q^{ROI} is the maximum influence P^* has in Q, not all $q_j \in Q^{ROI}$ will be influenced by the presence of P^*. Some configurations may lie within viewing distance of P^* but not observe any $p_k^* \in P^*$. If $q_j \in Q^{ROI}$ do not collide with P^*, when revaluating their coverage v^* it should be equivalent to v. As the same coverage is attained, the unique coverage of these configurations will ensure they remain in the updated tour T^*.

A conceptual example of the influence of P^* has on Q and P is presented in Figure 7-4. For any given offline tour T (Figure 7-4a), the introduction of P^* creates a ROI around each p_k^*. Projecting the geometrical ROI into the Q captures a set of configurations that are considered under the influence of P^* (Figure 7-4b). To ensure $P^{\#}$ is observed at the conclusion of each sampling phase, P^{ROI} is formed to encapsulate both $P^{\wedge}$ and P^* (Figure 7-4c).

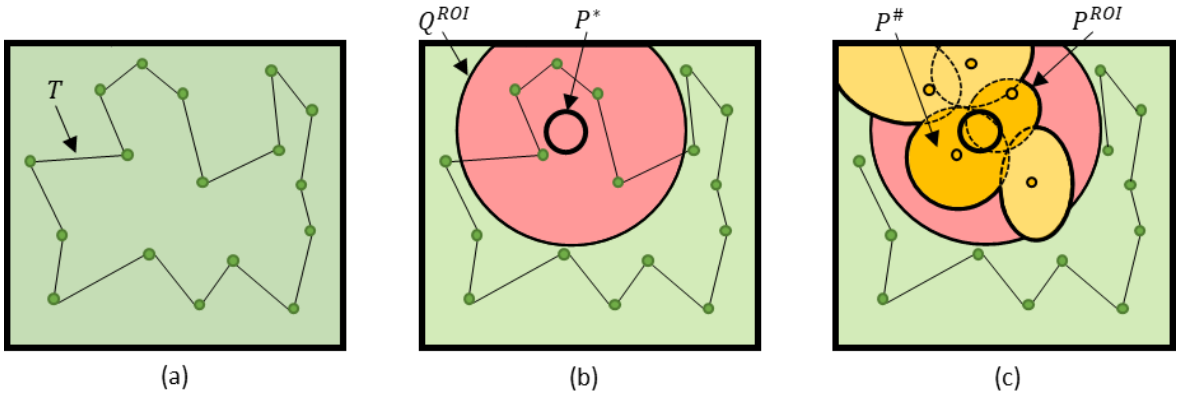


Figure 7-4: A conceptual representation of a region of interest (ROI) used to create Q^{ROI} and P^{ROI}. (b) The introduction of P^* creates Q^{ROI} within Q to segment T. (c) After segmenting T using Q^{ROI}, P^{ROI} can be found through including all of Q^{ROI} coverage. The example highlights all $P^{\#}$ within P^{ROI} to ensure all primitives of P^{ROI} remain covered.

7.2.3 Compromised Robot Configuration and Occluded Primitives

As described in Section 7.2.2, Q^{ROI} and P^{ROI} have been defined as the subsets in Q and M respectively. Within these sets there potentially exists set of compromised configurations and occluded primitives due to the influence of P^*. To capture the set of configurations that can no longer be physically achieved and the primitives that can no longer be observed by the robot, two sets, $Q^{COMPROMISED}$ and $P^{OCCLUDE}$ are defined as subsets of Q and M respectively to keep track of compromised configurations and occluded primitives over the lifetime of the inspection. As P^*, determines if configurations are unreachable or if primitives become occluded, $Q^{COMPROMISED}$ and $P^{OCCLUDE}$ only exist within Q^{ROI} and P^{ROI}, respectively.

7.2.4 Environmental and Mapping Influenced Prison Cells

Prison cells encountered when planning offline were assumed to be only caused by environmental constraints (Section 4.2.2). Since offline maps of the environment are assumed to be of high-quality, no additional constraints are added to the established environmental constraints. In the online case, *prison cells* can also be formed due to mapping limitations that occur in addition to the environmental constraints. This creates two types of *prison cells* that can form within the environment, *environmental prison cells* and *mapping prison cells*.

Mapping prison cells differ from *environmental prison cells*. Unlike an *environmental prison cell* that traps valid configurations within the actual geometry of the space, *mapping prison cells* trap primitives due to an insufficient understanding about the geometry of the space. As the map evolves over time, new features can completely occlude existing

primitives of the original map. In reality, these primitives that become occluded, have always been occluded and no longer represent actual observable geometries within the map. Figure 7-5 illustrates the differences between *environmental* and *mapping prison cells* due to online map updates.

Maintaining a complete map of the environment at all times requires the mapping system to remove trapped primitives. However, determining when primitives should be removed in an evolving environment is a challenging problem. Without a complete understanding about the environment, it is unknown as to whether occluded primitives are due to *environmental*

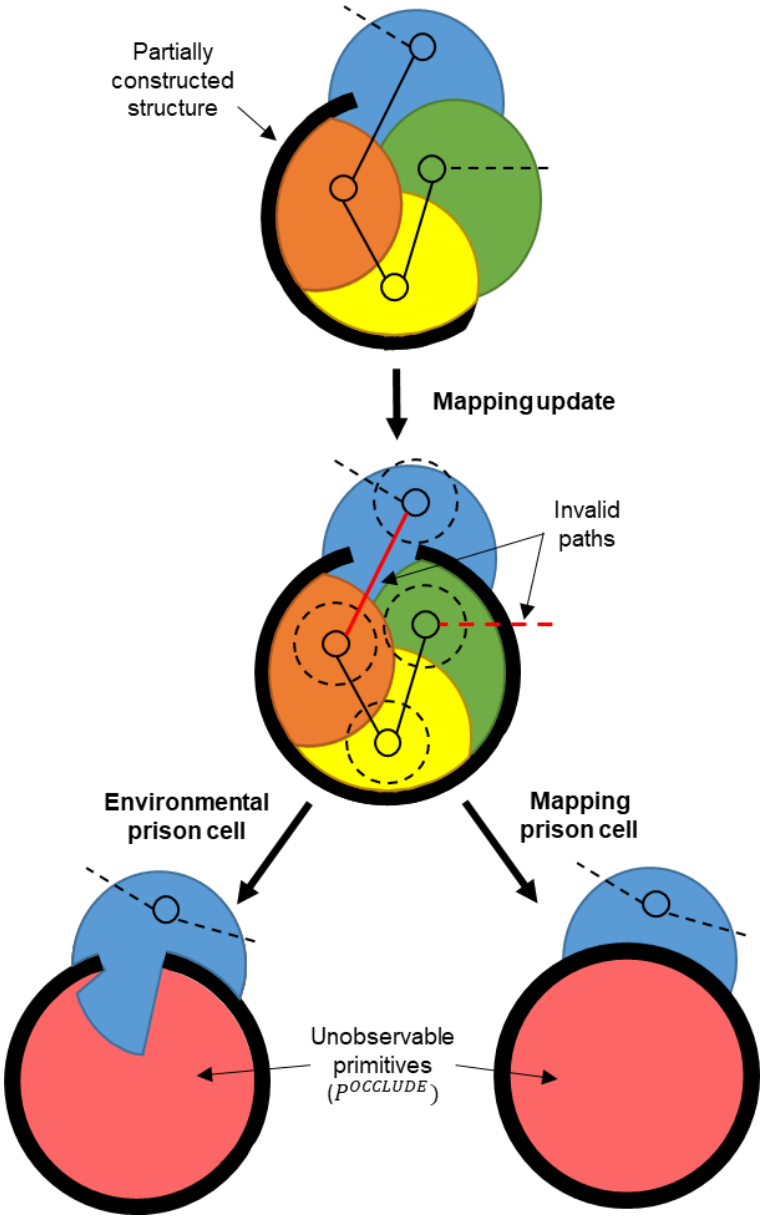


Figure 7-5: Evolving structures in the environment create either an *environmental* or *mapping prison cell* that will invalidate the existence of a path to surrounding configurations outside the cell. As these prison cells are difficult to identify during execution, the unobservable primitives trapped by these either of these prison cell types will be treated equally by the *adaptive coverage planner*.

or *mapping prison cells*. Attempting to remove primitives as the map evolves may prove to be a difficult exercise to undertake online.

The worst-case scenario is where the mapping system removing occluded primitives that are caused by *environmental prison cells*. These primitives are strictly unobservable due to limitations of the robotic platform and environmental constraints and must remain in the final map of the environment. Only when a complete map has been formed, can the removal of the occluded or trapped primitives be determined. As a result, the mapping system will not remove any primitives during execution. The mapping system will maintain the original primitive indexing of the original map and will only introduce new primitives to the planning problem (*Assumption 3*; Section 3.6.1).

To the *adaptive coverage planner*, *environmental* and *mapping prison cells* result in the same outcome. As trapped primitives are still included in the planning problem, they are still considered observable and will be required to be sampled. As a result, the coverage planner will continue to generate configurations to observe trapped primitives which in reality do not exist. Until these primitives are deemed unobservable, configurations generated for these primitives are deemed to be valid but will be rejected on the basis of not being feasibly reached. This situation is identical to how *environmental prison cells* influence configurations (Section 4.2.2).

As configurations generated within either prison cell type cannot be feasibly reached, the *adaptive coverage planner* does not need to determine what type of prison cell is the cause and can handle these situations identically. Therefore, whilst these *prison cells* are generated due to two different reasons, any trapped configurations or primitives that are removed from the planning problem are added to $Q^{COMPROMISED}$ and $P^{OCCLUDE}$ respectively. Sections 7.4.2, 7.5.2 and 7.6.2 discuss how *environmental* and *mapping prison cells* are captured in both the sampling and planning phases of the online planning problem.

To determine if a primitive was excluded from the planning problem due to an *environmental* or *mapping prison cell*, is evaluated after execution. As $P^{OCCLUDE}$ records all unobservable primitives, primitives removed by the mapping system, results in only $P^{OCCLUDE}$ containing unobservable primitives due to *environmental prison cells*. Since *mapping prison cell* primitives are removed, the actual coverage achieved by the coverage planner can be reported.

7.3 The Adaptive Sampling-based Coverage Planner

Figure 7-6 presents the proposed adaptive sampling-based coverage planning architecture. The *adaptive sampling-based planner* consists of three phases;

1) ROI Validation,
2) Online CSP, and
3) Online MPP.

The ROI Validation phase is responsible for utilising ROIs to determine which configurations of the current plan T are either preserved or candidates for replanning. The Online CSP uses the functionality of the offline CSP and SCP procedures to generate coverage for new structures detected by the mapping system. Finally, the Online MPP is responsible for replanning and merging the new paths into the current plan using the offline MPP algorithm. This iterative algorithmic process allows the robot to continually adapt its plan online while preserving as much of the uninfluenced plan as possible without compromising coverage.

By maintaining a decoupled approach between sampling and path planning methods, the architecture of the online coverage planner generates a planning problem that is indistinguishable to the offline CSP and MPP methods. A *redundant roadmap* is created over a subset of the environment that is yet to be covered and the *lazy point-to-point planner* (LPP) iteratively solves individual sub-plans for each ROI between designated start and finishing locations.

In the following sections, the three phases of the *adaptive sampling-based coverage planner* are discussed.

Section 7.4: Phase 1) ROI Validation (segmentation)
Section 7.5: Phase 2) Online CSP (replan coverage)
Section 7.6: Phase 3) Online MPP (replan paths and merge)

Each section also highlights amendments made to the offline planning procedures discussed in Section 4.2 to enable the functions to perform robustly in an online situation.

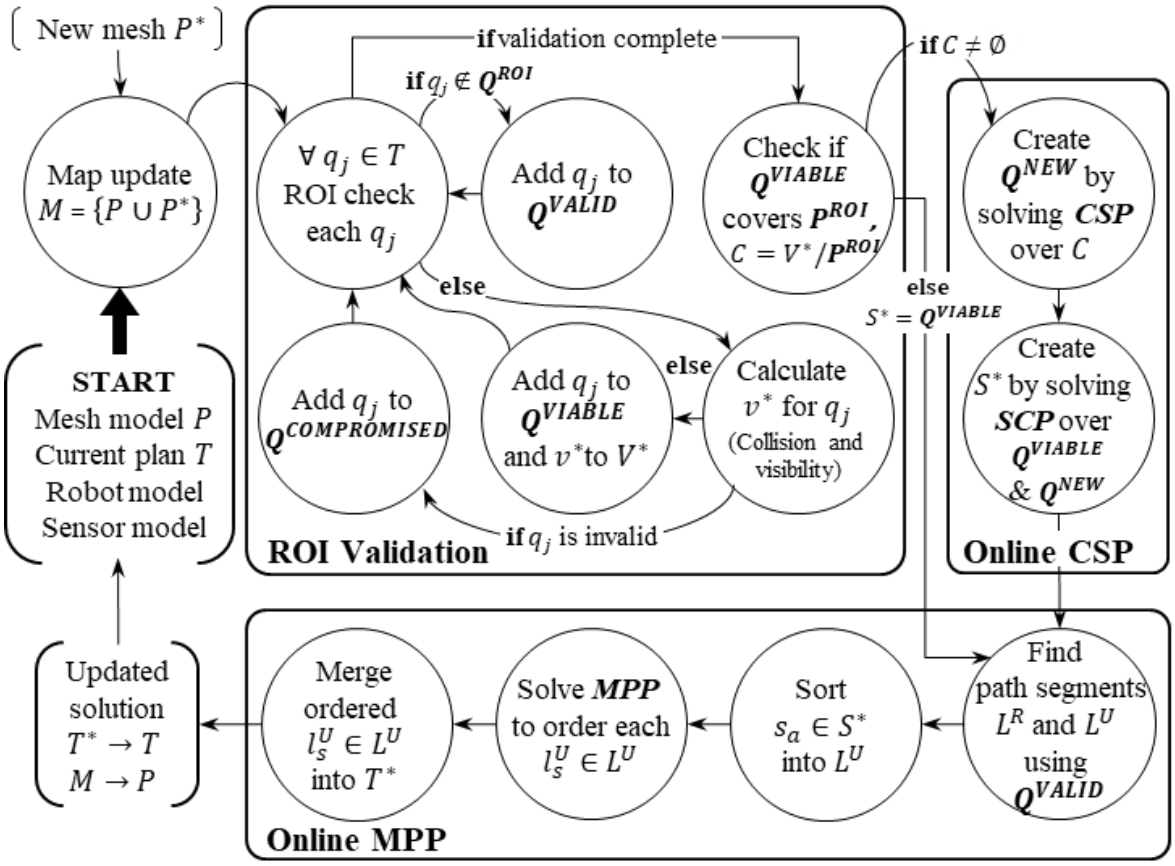


Figure 7-6: Flowchart representation of the *adaptive sampling-based coverage algorithm* using a *plan repair strategy*. The *adaptive sampling-based coverage planner* includes an initial validation step that determines if the existing configurations reside inside or outside the *region of interest* (ROI). The separation allows the *adaptive coverage planner* to only replan regions within ROIs.

7.4 ROI Validation

Before the new features are covered by new configurations and added into a new inspection plan T^*, the current plan T needs to be segmented based on the ROI to determine the influence of P^*. The ROI Validation, uses the definition of Q^{ROI}, to iteratively check to determine which $q_j \in T$ to preserve (Q^{VALID}), to revaluate (Q^{VIABLE}), or remove ($Q^{COMPROMISED}$). Figure 7-7 illustrates the ROI Validation procedure, presented in Figure 7-6.

All $q_j \notin Q^{ROI}$ create the preserved configuration set Q^{VALID}. As Q^{ROI} determines which configurations exist outside the influence of P^*, the elements of the Q^{VALID} are all assumed to be free of collision and still observes the same primitives as originally calculated in the offline plan. However, as P^* introduces new primitives, it may be possible for $q_j \in Q^{ROI}$ to collide with or observe primitives of P^*.

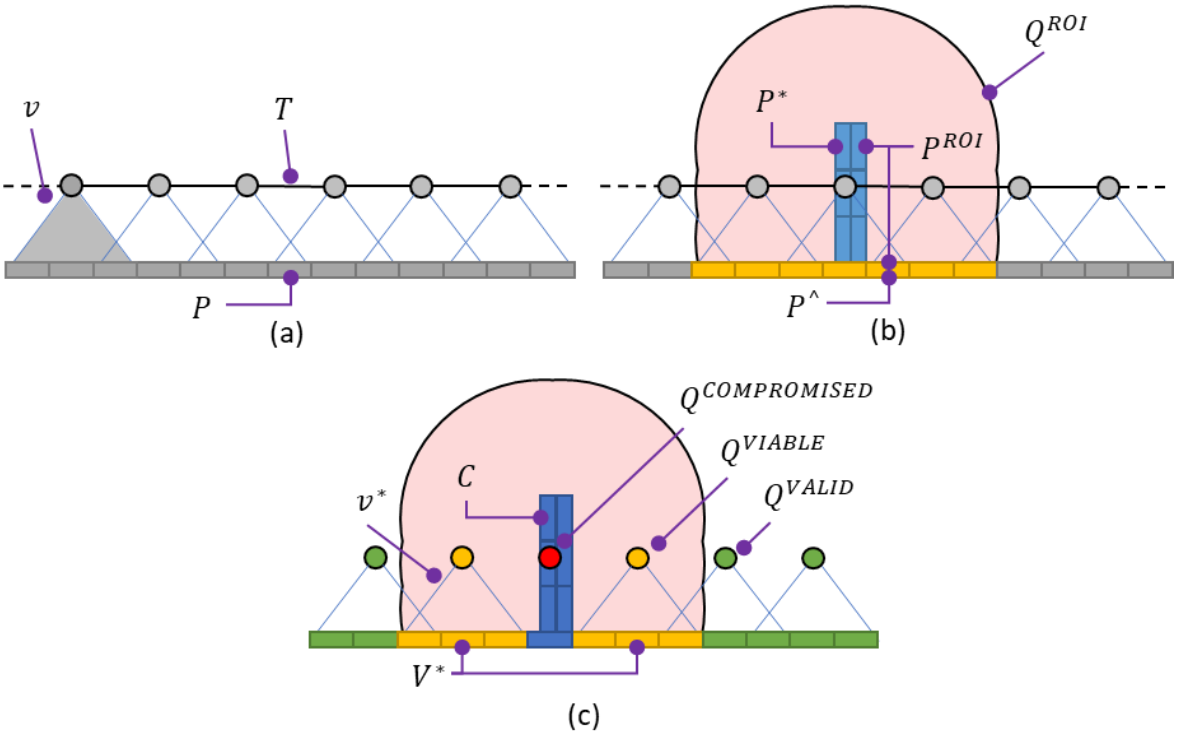


Figure 7-7: Passing the existing tour through the *region of interest* (ROI) Validation phase. (a) A segment of the current tour. (b) The new primitives (light blue) bounded by the ROI (red). (c) The current tour is evaluated to determine the current status in and around the ROI.

As illustrated in Figures 7-6 and 7-7c, to preserve as many configurations in T, each $q_j \in Q^{ROI}$ is checked for collision and previous coverage revaluated. The evaluation first checks the collision status of the robot to ensure the inspection pose can still be achieved. As there is a possibility that $q_j \in Q^{ROI}$ may observe P^*, the original v is recalculated s.t. $v^* = \{(p_i \subseteq v) \cup (p_k^* \subseteq P^*)\}$. Configurations that pass these evaluations are added to Q^{VIABLE} with each v^* collectively recorded in V^* to determine later if all P^{ROI} is covered. Configurations that fail validation are added to $Q^{COMPROMISED}$ and removed from the planning problem.

After all $q_j \in T$ have been evaluated, Q^{VIABLE} is checked to determine if all the configurations cover all elements of P^{ROI} s.t. $= V^*/P^{ROI}$, where C is the residual unobserved primitives not seen by V^*. If the coverage of Q^{VIABLE} is not complete over P^{ROI} ($C \neq \emptyset$), C is passed along with Q^{VIABLE} to the Online CSP. If Q^{VIABLE} covers all P^{ROI} ($C = \emptyset$), the requirement for complete coverage is held and no q_n^* are required. Therefore, the Online CSP is skipped with Q^{VIABLE} accepted as the covering set S^* of P^{ROI} and proceeds to the Online MPP to revaluate the paths within Q^{ROI}.

7.4.1 Computation of the ROI

To determine if q_j is an element of Q^{VALID} or Q^{VIABLE} relies on whether q_j is within the local neighbourhood of p_k^*. To minimise the computation for evaluating each $p_k^* \in P^*$ to build Q^{VIABLE}, a nearest neighbours check is used to determine if q_j is within the local neighbourhood of p_k^*. Since the mapping system only adds new primitives to a planning problem (*Assumption 3*), a threshold index (m_{index}) is provided to the *adaptive coverage planner* by the mapping system to delineate the entries of the new primitives (P^*) from the existing primitives (P) of M. If any primitive index returned from the nearest-neighbours search is higher than m_{index}, places the configuration within the ROI and subsequently a part of Q^{VIABLE}.

The calculation of C is performed by reassessing the coverage v of $q_j \in T$ the beginning of each replanning iteration. The *adaptive coverage planner* initialises all primitives of M to be unobserved and consequently places them to an *unobserved primitive list* (Section 4.2.3). When validating each $q_j \in T$, primitives of v and v^* for every $q_j \in Q^{VALID}$ and $q_j \in Q^{VIABLE}$ are removed from the *unobserved primitive list*. The remaining primitives in the *unobserved primitives list* after evaluation become C. As the remaining primitives in this list were not observed by any $q_j \in T$, ensures that $P^{\#}$ and primitives P^* will always be included in C.

As this thesis works with large sized meshes, an AVL tree was used to represent the *unobserved primitive list*. Continuous calculation of C requires inserting, searching and removing elements in list and an AVL tree provided a suitable structure to perform these functions quickly and efficiently (Puntambekar, 2009).

7.4.2 Revaluating the Trapped Configuration Heuristic due to Partial Map Updates

When planning is conducted offline, any configurations generated inside fully enclosed objects or *prison cells* are removed using the *Trapped Configuration Heuristic* (TCH) which performs a ray-trace to the environment's exterior boundaries and counts the number of intersections against the layers in the mesh manifold (Section 4.2.2, Figure 7-8a). When planning online, with an evolving mesh, the TCH is no longer guaranteed to hold.

Figure 7-8b illustrates a simple example of an incomplete object and how the TCH no longer holds for configurations located in free space. To ensure trapped configurations continue to be captured, so *prison cells* are avoided, the intersection test, that requires at least two ray

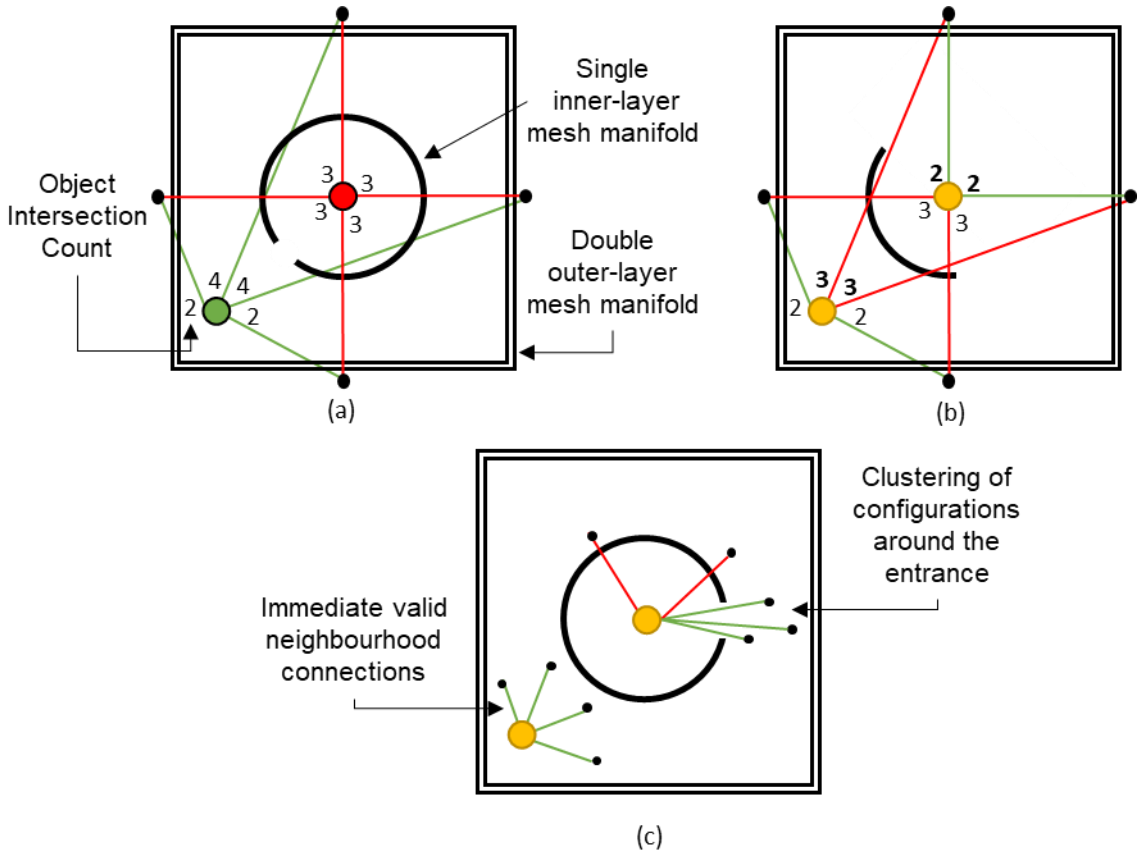


Figure 7-8: Revisiting the trapped configuration problem. (a) A complete environment model can handle the strict ray tracing constraint. (b) Relaxing the intersection count constraint enables configurations to exist within partially constructed environments. (c) Clustering around the entrances of an object can cause the Nearest Neighbours Heuristic to pass despite the robot being physically unable to pass through the gap of the object.

traces to pass, is relaxed. Therefore, reassessing the example in Figure 7-8b, the *relaxed* TCH (R-TCH) allows both configurations to remain valid within the environment. As it is indeterminate as to when the feature will be fully constructed, the R-TCH does not hold for all cases. Therefore, an additional heuristic is used to determine if a configuration is *potentially trapped.*

A *Nearest Neighbour Heuristic* (NNH) is added to examine the direct connectivity between *potentially* trapped configurations and nearby configurations. As generating a motion plan between all the immediate neighbours of the *potentially trapped configuration* would be an expensive exercise, a ray trace to the immediate neighbours is used as a proxy. If a direct line of sight between two or more configurations is achieved, indicates that a path may exist between '*potentially'* trapped configurations and the configuration in the tour. As the environment is not considered when evaluating the NNH, giving the *potentially trapped configuration* upwards of ten neighbours is suitable to determine if the configuration is potentially reachable.

The development of the NNH was derived from the highly compact covering sets, as seen in the benchmark experiments (Section 4.4). The majority of the paths are solved using a direct path and given the robot planning constraints, the tank environments are expected to produce a covering set into the thousands with closely compact neighbours. However, given the nearest neighbours heuristic is a proxy for a motion plan, and no collision checking along the approximated path is performed, there is a caveat to this approach. When the structures begin to form, the NNH has the potential to pass despite the robot being physically unable to pass through the object. In highly compact covering sets, a crowding of configurations can occur around the entrances of an object, allowing the naïve nearest neighbours check to pass (Figure 7-8c). A trapped configuration that passes these tests in the ROI Validation or Online CSP phases creates a *prison cell* in the Online MPP phase. To compensate for the case that both the R-TCH and NNH pass a *trapped configuration*, the functionality of the LPP has been extended to remove these configurations during the path planning phase (Section 7.6.2).

Despite the occasional chance that a *trapped configuration* is passed the LPP, both R-TCH and NNH in tandem, create a suitable and computationally efficient check to determine if a configuration is trapped without having to calculate a motion plan. When the new structures begin to take shape, more configurations will be captured by these heuristics. Any configurations that are trapped are added to $Q^{COMPROMISED}$ and the unique primitives of these configuration remain in C to be covered again.

7.5 Online CSP

The Online CSP is responsible for creating the covering set Q^{NEW} to observe C (Figure 7-9). The random sampling procedure, as utilised in the offline algorithm with an amended *optimal sampling procedure* (Section 7.5.1), is used to generate a *redundant roadmap* to cover C instead of P. The SCP is applied over Q^{NEW} and Q^{VIABLE} to produce a feasible covering set S^* that covers all primitives in P^{ROI}. Any members of Q^{VIABLE} that are discarded from the greedy and pruning procedures for no longer contributing significant or unique coverage are added to $Q^{COMPROMISED}$.

7.5.1 Optimal Sampling Procedure

Random sampling is suitable in the online planning situation due to minimal assumptions random sampling places upon the environment, therefore, making it suitable to employ over

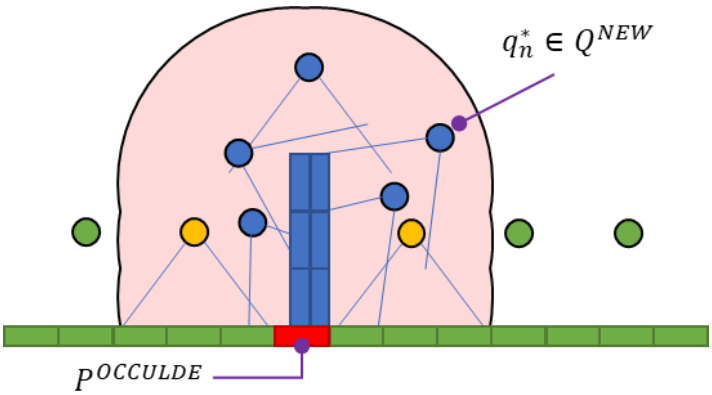


Figure 7-9: Sampling new configurations across the unobserved primitive set C. A new *redundant roadmap* is generated over the unobserved primitives (blue) within the *region of interest* (ROI). The two existing configurations (yellow) are retained for their unique coverage. A single primitive is partially occluded by the new structure (red) and is excluded from the coverage.

incomplete objects. However, random sampling does not guarantee the configuration contains the best quality observation. Randomly sampled configurations can generate viewing angles that are oblique to the surfaces. In some circumstances, samples generated with oblique angles can be the best attempt to view the complex surfaces. However, when sampling is performed over simpler surfaces, a more deterministic sampling process that maximises the coverage over these surfaces may be more beneficial. A deterministic sampling approach will create regularity in positioning which will aid in maintaining consistent image quality. In an inspection task dependant on capturing high-quality images, maintaining a level of consistency across all images is the preferred outcome.

To reduce the number of random samples used to cover the environment, Englot and Hover (2012b) segmented the environment into simple and complex surfaces. Sweep paths generated from regularly spaced waypoints were used to cover the large flat surfaces from a more ideal viewing distance while random sampling was left to cover the remaining complex surfaces. However, since the segmentation process differs between environments and the sweep paths are 2D projections over the flat surfaces, the sweep path technique is not directly applicable to the online planning problem or the target environment. Therefore, instead of introducing any additional pre-processing step to enable systematic sampling over particular surfaces, the existing sampling process to create the *redundant roadmap* is modified to first seek the creation of an *'optimal'* viewing position before reverting to random sampling to complete the remaining redundancy of a primitive.

When random sampling configurations within the local neighbourhood, there is a viewing position that would be considered the *optimal position* to view the surface in relation to

image quality. Generally, over a flat surface the *optimal position* is directly above the surface, as it allows the full exploitation of the viewing angle across the surface (Figure 7-10a). While the *optimal position* is subjective to the application, a suitability placed 'optimal' position can remove several random configurations from being selected solving the SCP (Figure 7-10b). As the parameters to define the *optimal* viewing distance is based on the intrinsic sensor constraints, no parameterisation or pre-processing of the environment is needed to include a check on the *optimal* viewing location of a selected primitive.

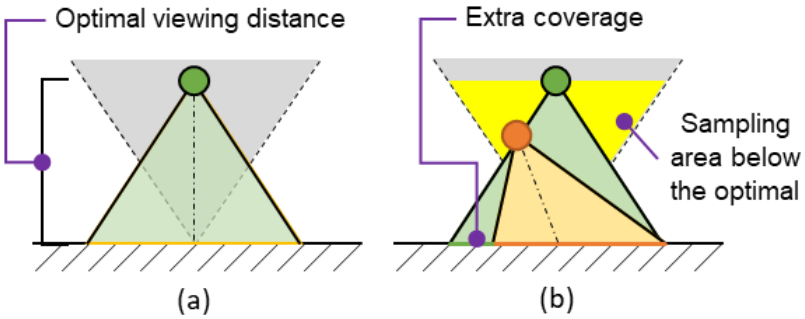


Figure 7-10: Optimal Sampling Procedure. (a) Sampling above the surface at the *optimal position* allows for (b) a better observation of the surface than a large percentage of random samples, especially below the *optimal position*.

A change to the sampling procedure to create the *Optimal Sampling Procedure* (OSP), allows for the *optimal position* to be sampled first in an attempt the gain the best possible coverage of the selected primitive before proceeding to use random samples to view the primitive again. If the *optimal position* is not valid due to collision or occlusion, the primitive's *primitive rejection count* (Section 4.2.3) is not increased. The *primitive rejection count* is only applied failed random samples.

When solve the SCP over the *redundant roadmap*, *optimal* and random configurations are treated equally. As the greedy algorithm is configured to take configurations containing the highest number of primitives, if a set of *optimal* configurations contains more coverage than a random configuration viewing the same primitive, it will be accepted into S^*. It is therefore assumed that in areas of local planarity, *optimal* configurations would dominate over random samples, and in areas of restrictive geometry, random samples will dominate.

Another benefit of regularly spaced configurations is that it aids in the production of smoother paths. As the sampling and path planning process are separate, the resulting tours are disjointed as the piecewise path connections are joined together. The accumulation of paths does not produce successive smooth trajectories that would be beneficial for a robot

possessing complex motion planning constraints. As a result, the energy consumption and motion planning along these trajectories would be more expensive compared to trajectories that are noticeably straighter. While, this thesis does not focus on the energy expenditure of a coverage plan, promoting the use of *optimal positions*, where possible, will aid in creating coverage plans that are easier to solve for the high-fidelity motion planner.

7.5.2 Tracking Occluded Primitives over the Lifetime of the Inspection Task

In Section 4.2.3, a user-defined *primitive rejection limit* was introduced to limit the number of times a primitive can fail being observed before the primitive is deemed to be *unobservable*. Given there is an increased chance of *prison cells* within an evolving environment, the user-defined *primitive rejection limit* is critical to ensure occluded primitives are captured and removed as early in the sampling process as possible. Constant sampling over occluded primitives will result in an increase in sampling time that may not contribute any further coverage. As the *adaptive coverage planner* cannot distinguish between *environmental* and *mapping prison cells*, all primitives exceeding the rejection count are added to $P^{OCCLUDE}$.

Once a primitive becomes a part of $P^{OCCLUDE}$ after sampling is complete, the primitive will no longer be considered a part of P^{ROI}, if present within a ROI, in the future planning updates. Primitives of $P^{OCCLUDE}$ are immediately removed from the *unobserved primitive list* at the beginning each planning iteration. Primitives once added to $P^{OCCLUDE}$ can only be removed if observed indirectly by another configuration. As the mapping system only adds new primitives to the planning problem, it is unlikely that once a primitive is added to $P^{OCCULDE}$ it will be observable in future iterations. Ensuring that all occluded primitives are maintained over the lifetime of the inspection task will aid in determining the actual cause of occlusion when analysed by the mapping system post-inspection.

7.6 Online MPP

The final stage of the new *adaptive coverage planner* is to re-assess the validity of the paths within Q^{ROI} and merge S^* into an existing plan using the Online MPP as shown in Figure 7-11. The role of the Online MPP determines the preserved paths throughout the environment, finds the best location for the new and original configurations to be added back into the tour and solve the paths within Q^{ROI} to produce an updated tour T^*. The functionality of the Online MPP procedure attempts to reduce the overall computation

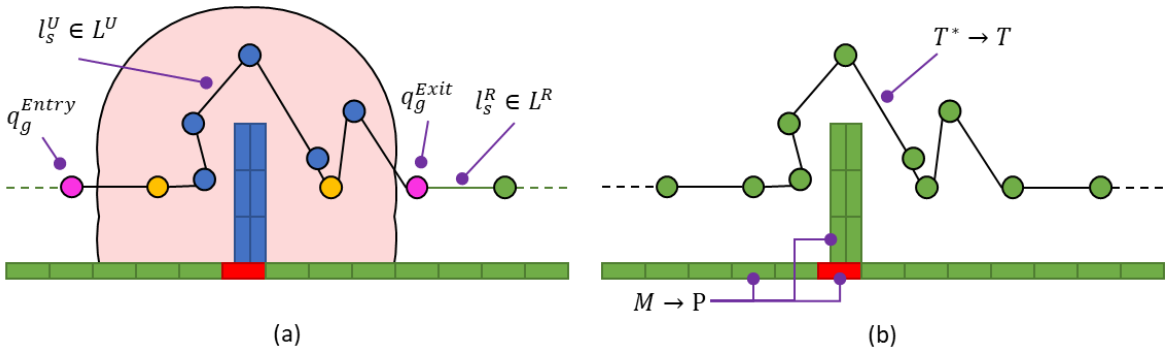


Figure 7-11: Replanning path segments back into existing plan. (a) The path segment is replanned between the entry and exit gates into around the region of interest. (b) The tour and map are updated to reflect the new paths and coverage determined in the current iteration. The next iteration will exclude the unobserved primitives completely (red) if they are not included within the *region of interest*.

compared to the *full replan strategy* by preserving as much of the original tour T as possible and solving smaller MPPs.

This problem of reducing the replanning effort is facilitated by breaking T into path segments. When generating an offline T, there exists only one path segment from start to finish. However, when replanning occurs online, T may intersect multiple ROIs. To distinguish between paths that are either outside or within the ROI, T is divided into multiple path segments. Separating T into path segments, allows segments outside the ROI to be preserved and those intersecting the ROI, to be replanned. That is, $T^* = \{L^R \cup L^U\}$, where

L^R is the set of *resolved path segments* l_s^R outside the ROI and L^U is set of *unresolved path segments* l_s^U inside the ROI.

The *resolved path segments* are constructed from stepping along T, and adding to l_s^R all $q_j \in Q^{VALID}$ until q_j encounters an entry gate q_g^{Entry}. A q_g^{Entry} is the last $q_j \in Q^{VALID}$ before entering the ROI, indicating that the path between $q_j \in Q^{VALID}$ and $q_{j+1} \in Q^{ROI}$ will need to be replanned. Equation 7.3 defines the set of all entry gates Q^{Entry}.

$$Q^{Entry} \triangleq \{q_j \in Q^{VALID} | (q_j \in Q^{VALID}) \wedge (q_{j+1} \in Q^{ROI}), \forall q_j \in T\} \quad (7.3)$$

Conversely, the path from $q_{j-1} \in Q^{ROI}$ to $q_j \in Q^{VALID}$ will indicate the departure of the ROI and will create an exit gate q_g^{Exit}. New l_s^R will begin construction again upon encountering the next $q_j \in Q^{VALID}$ in T. Equation 7.4 defines the set of all exit gates Q^{Exit}.

$$Q^{Exit} \triangleq \{q_j \in Q^{VALID} | (q_{j-1} \in Q^{ROI}) \wedge (q_j \in Q^{VALID}), \forall q_j \in T\} \quad (7.4)$$

Unresolved path segments are constructed from the pairing of ROI entry and exit gates and serve as a path segment to plan into T^* subsets $s_a \subset S^*$. Equation 7.5 defines the set of all unresolved path segments L^U.

$$L^U \triangleq \{Q^{Entry} \cup s_a \subset S^* \cup Q^{Exit}\} \tag{7.5}$$

Elements of S^* are sorted into each l_s^U by evaluating the point-to-line distances between each element of S^* between each pair of q_g^{Entry} and q_g^{Exit}. When all $l_s^U \in L^U$ are created, each $l_s^U \in L^U$ creates an unordered ROI sub-plan that can be solved by the LPP. Assuming these sub-plans are sufficiently small in size, will reduce computation of the problem compared to solving the MPP across the whole space which would be carried out in the full replan case.

In the case where $q_{j+1} \notin Q^{ROI}$ resulting in no Q^{Entry} and Q^{Exit}, all elements of S^* are collectively solved as a singular sub-plan between the last and second Q^{VALID} in T. An example of this situation occurs when the new surfaces and the ROI do not intersect T. While a rare occurrence, especially in highly clustered coverage sets, providing this extra clause ensures the new coverage is always added into the plan.

As the path segments of T are solved in tour-order, upon solving for each $l_s^U \in L^U$ the result can be merged together with L^R to form T^*. The updated solution is provided to the motion planner to update the new paths before being supplied to the robot.

Before the next M is supplied, the current $M \longrightarrow P$ and $T^* \longrightarrow T$. This allows the planning iteration to replan knowing everything beforehand has been resolved. This process continues until the robot has completed the tour and no further map updates are provided.

7.6.1 Solving Each ROI Sub-plan

To ensure the TSP solver finds a solution between specified start and finish positions, each sub-plan is solved using an algorithm for finding the shortest *Hamiltonian path* by introducing a *dummy* city to constrain the TSP solution (Applegate et al., 2011). Forcing the TSP to start and end at different endpoints has also been known as the *Travelling Salesman Sub-problem* (Englot and Hover, 2012b) or *Shortest Sequencing Problem* (Alatartsev, Stellmacher, and Ortmeier, 2015). Solving the *Travelling Salesman Sub-problem* has a few different solutions. Englot and Hover (2012b) ensures the entry and exit terminals of the sweep path appear adjacent in the final TSP solution by assigning a zero cost between entry

and exit terminals.

The approach used in this thesis introduces a *dummy* city, or configuration, to augment the adjacency matrix to ensure the start and finishing locations are adjacent in this final inspection plan (Figure 7-12; Applegate et al., 2011). A zero cost is assigned to the edges connecting the starting and finishing locations with the *dummy* configuration. To ensure no other path is viable to the *dummy* configuration, a value that is significantly larger than any other connection is assigned to all other configurations to the *dummy* configuration (Figure 7-12b(i)).

In the final solution, the *dummy* configuration can be removed, placing the start and finish position of each sub-plan adjacent to each other (Figure 7-12b(iii)). As the TSP solution is assumed to be symmetrical, separating the tour at the *dummy* configuration allows a TSP sub-problem to create a tour starting from q_g^{Entry} and ending at q_g^{Exit}.

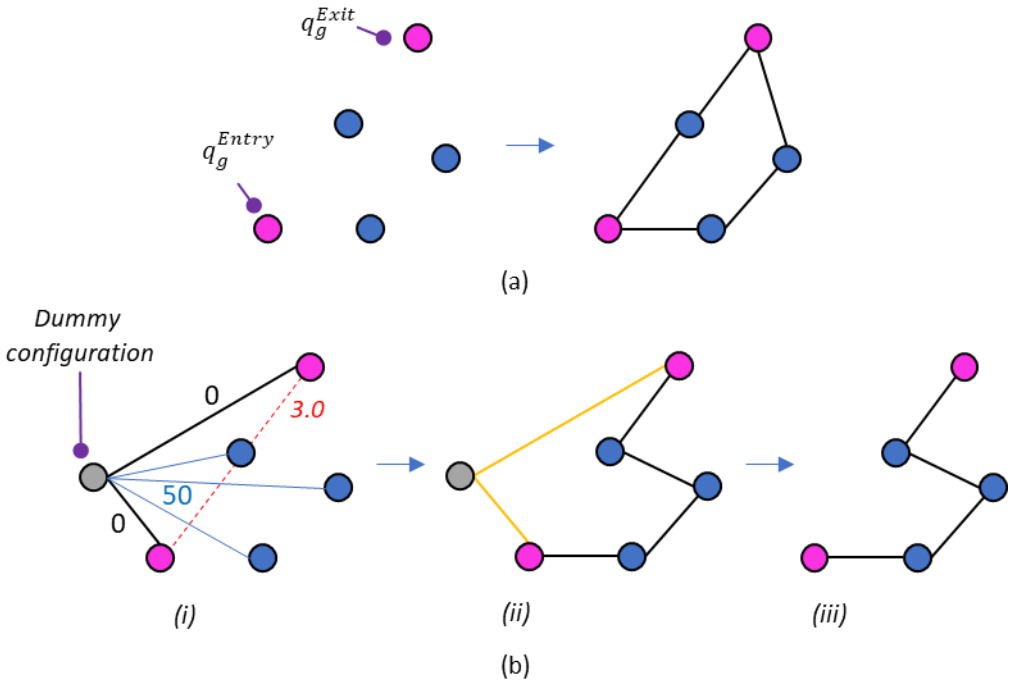


Figure 7-12: The Travelling Salesman Sub-Problem. (a) A standard *travelling salesman problem* (TSP) solution starting and finishing at the entry gate. (b) A solution using a dummy configuration to solve the TSP sub problem. (b-i) Assigning a zero-edge cost from the start and finish to the dummy configuration (black) with a large edge cost from the dummy configuration to all other configurations (blue). This value needs to be greater than the largest edge cost in the problem (red) to avoid connecting any other configurations to the dummy configuration (b-ii) The final solution forces the dummy configuration to be adjacent to the gate pair (orange). (b-iii) Removing the dummy configuration orders a TSP Sub-Problem starting and finishing at the gate pair.

7.6.2 Enabling the LPP to Remove Trapped Configurations due to Prison Cells

There is the possibility that the heuristics developed to capture *trapped configurations* fail (Section 7.4.2). When this happens, new paths to this configuration are potentially infeasible. Figure 7-13 illustrates a simple example of how no RRT can find a solution through the opening of the object to connect to the *trapped configuration*.

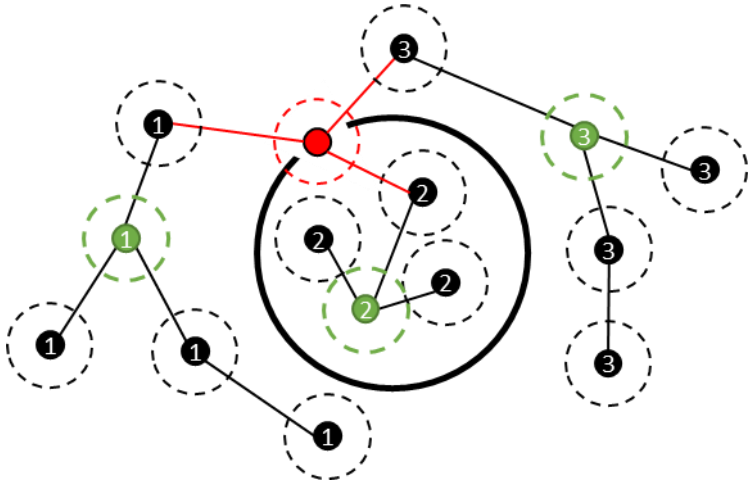


Figure 7-13: Example of a trapped configuration within a partially constructed object. RRTs attempt to connect to configurations 1-2 and 2-3 however collision checks on RRT paths on the entry of the object (red) impede these paths from connecting to the trapped configuration (2). Configuration 2 is deemed to be trapped and is removed from the tour.

In the LPP, if a configuration is trapped, a viable path to that configuration cannot be formed. If a viable path cannot be formed between two configurations for a given planning iteration constitutes an invalid tour. An invalid tour in an iterative replanning system will not allow the robot to effectively conduct the inspection mission. As these situations are difficult to predict, it is acceptable to remove *trapped configurations* in the path planning stage to allow a valid solution to exist providing the removed configurations and the associated coverage is reported as unreachable and unobservable respectively.

The current implementation of the LPP does not remove *trapped configurations*. While majority of the *trapped configurations* are expected to be captured in the ROI Validation stage, a *trapped configuration* under the current LPP implementation will not produce a successful tour; a consequence that is not acceptable for online planning. Therefore, for the LPP to be flexible and consistently provide valid tours upon each planning iteration, an additional heuristic is added to the LPP to detect and remove *trapped configurations*. The *Trapped Configuration Heuristic for Path Planning* (TCH-PP) is used to provide the LPP with a removal process for *trapped configurations* so they do not produce invalid tours.

The TCH-PP is used to track the number of times a configuration fails to create an RRT paths to other configurations. Configurations that are trapped are unable to produce a path between the preceding and succeeding configurations in the tour. If a configuration fails to create a path to and from its current location in successive iterations, it is considered trapped and is removed.

Removing one *trapped configuration* is simple and can be removed in the first iteration. However, identifying and removing two or more configurations that are trapped within the same object requires more planning iterations to track the number of paths between configurations that fail. After additional planning iterations, the *trapped configurations* will continue to report failed paths to other external configurations. By cross-checking the failed attempts against failure reports from previous iterations, the most commonly failed configurations can be identified and removed.

Removing a configuration from the planning set will constitute an invalid planning iteration as the adjacency matrix needs to be reconstructed for the next planning iteration. Instead of reinitialising the matrix back to the Euclidean assumption, the adjacency matrix is reformed to include all previous solutions. As the planning problem has changed, reformatting the adjacency also requires reinitialising the best solution before the next replanning iteration.

Finally, whenever a configuration is removed it is important that the coverage that it once observed properly reflected with respect to the coverage of the overall environment. As the sampling process has already occurred for this planning iteration, the unique primitives that the *trapped configuration* observed is added $P^{OCCULDE}$.

7.7 Representative Concept Demonstration

Figure 7-14 illustrates conceptual demonstration of the *adaptive sampling-based coverage planner* implementing a *plan repair strategy* to provide coverage for pipe holder structure over an I-beam. Figure 7-14a overlays an offline plan calculated by the *offline sampling-based coverage planner* starting and ending at specific endpoints (green and red respectively) using the TSP sub-problem solution presented in Section 7.6.1. In Figure 7-14b, the pipe holder is introduced as the new feature P^* and is bounded within a representative cylindrical ROI that approximates the features influence on current tour T.

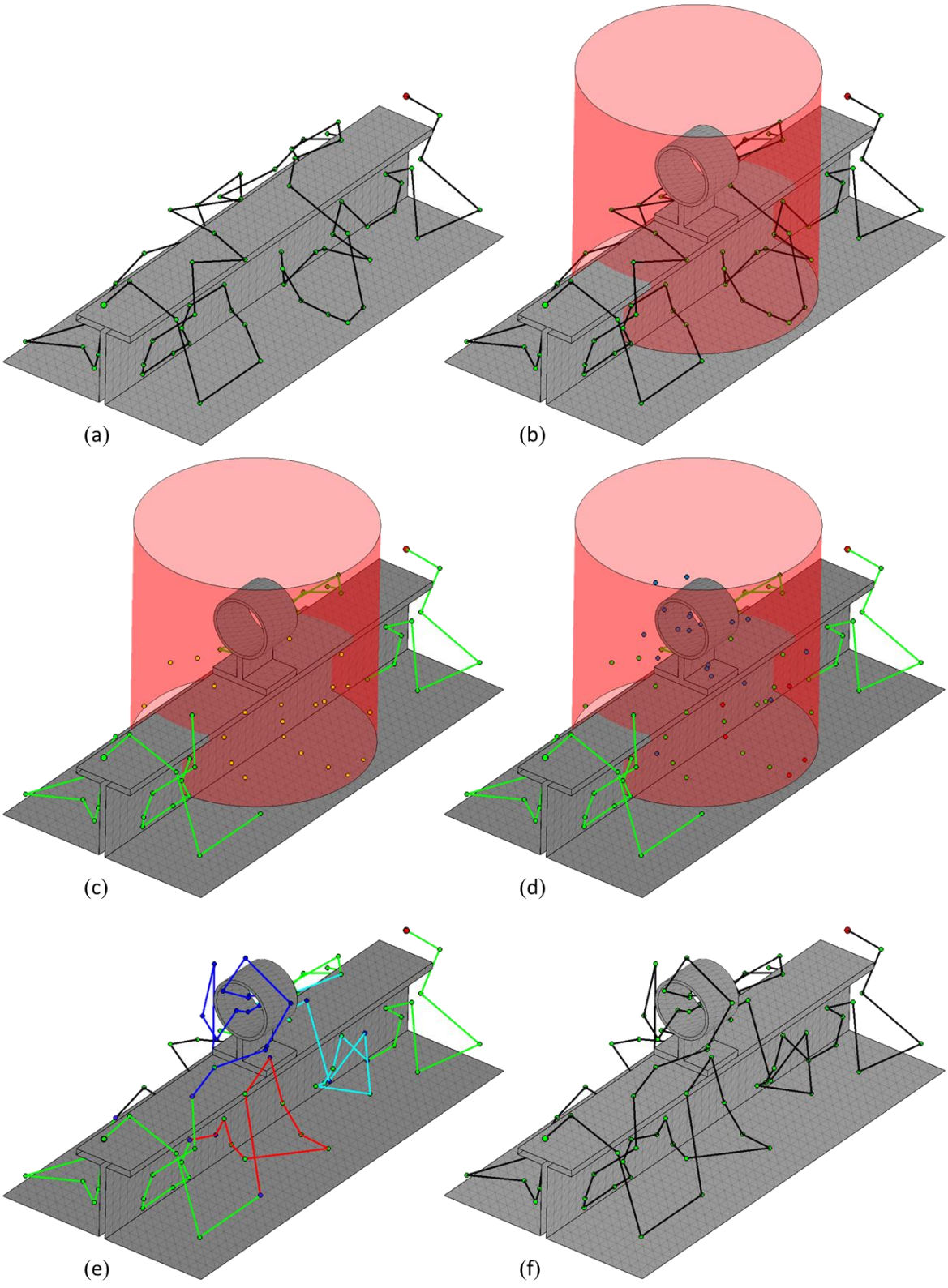


Figure 7-14: The adaptive sampling-based coverage planner replanning over a representative I-Beam structure. (a) Offline plan. (b) Introduction of a new structure bounded within an approximated *region of interest* (ROI). (c) ROI Validation. (d) Online covering sampling problem (Online CSP). (e) Online multi-goal planning problem (Online MPP). (f) Updated inspection plan.

Figure 7-14c demonstrates the first stage of the *adaptive sampling-based coverage planner* by performing cross checks over T to determine which configurations exist inside (Q^{ROI}) or outside the ROI (Q^{VALID}). Figure 7-14c also shows the *resolved path segments* (L^R), highlighted in green, which have been preserved outside the ROI. Configurations of Q^{VIABLE} are highlighted in yellow.

As the new feature is not covered by any original configurations, the Online CSP procedure samples new configurations and solves the SCP which combines both Q^{VIABLE} and Q^{NEW} to find S^* over P^{ROI}. The result of the Online CSP process is shown in Figure 7-14d with the accepted Q^{NEW} shown in blue and the surviving configurations of Q^{VIABLE} highlighted in green. Compromised configurations ($Q^{COMPROMISED}$) are highlighted in red.

Figure 7-14e demonstrates the Online MPP process. The elements of S^* are sorted into *unresolved path segments* (L^U) and solved using the MPP between Q^{Entry} and Q^{Exit}. Each *unresolved path segment* that was solved by the LPP is highlighted as a different colour. In this example five path segments were resolved within the one collective ROI that surrounds the pipe holder. Figure 7-14f shows the final tour update.

7.8 Probabilistic Completeness of the Adaptive Sampling-based Coverage Planner

The *offline sampling-based coverage planner* was proven by Englot and Hover (2012a) to be probabilistically complete. Both the sampling and planning procedures were presented with failure bounds that coincide with previous proofs given in other sampling-based path planning algorithms (Kavraki et al., 1998; LaValle and Kuffner, 2001). All that is required is that *prison cell* geometries are not present in the environment.

Given the *adaptive sampling-based coverage planner* is derived directly from the fundamental CSP and MPP procedures of the *offline coverage planner* to solve smaller sub-problems should ensure the probabilistic completeness of these procedures still hold. However, since the *adaptive coverage planner* has to handle evolving environments and segment the plan to reduce the replanning costs, the impact of managing these problems has the potential to impact on the completeness of the final solution.

The first consideration is handling the *prison cells*. Probabilistic completeness for the offline CSP and MPP sub-problems are able to achieve 100% coverage provided *prison cells* can

be avoided (Englot and Hover, 2012a). As discussed in Section 7.2.4, transient *prison cells* are expected to occur, if not frequently, in an online planning scenario. Through a series of implemented countermeasures that restrict sampling over occluded primitives (Section 7.5.2) and remove trapped configurations from the planning problem (Sections 7.4.2 and 7.6.2), ensure *prison cells* also do not affect the probabilistic completeness of the *adaptive sampling-based coverage planner*.

The second consideration is to ensure every configuration and primitive under the influence of P^* is appropriately captured. The ROI is used to determine which configurations and primitives are influenced by P^*. Assuming that all of the primitives of P that become unobserved ($P^\#$) due to P^* are candidates for resampling, s.t. $P^\# \subset P^\wedge$ (Section 7.2.2), ensures that all primitives that do require to be replanned ($P^\#$) will not be excluded in the resampling process and therefore will remained covered.

As mentioned in Section 7.5, the Online CSP procedure is a direct utilisation of the offline CSP algorithm. Providing the mesh manifold exists in the same dimension as Q (this work is $\mathbb{R}^3$) and the mesh density is of sufficient resolution that at least a single primitive can be observed, the probabilistic completeness proof as presented in Englot and Hover (2012a) holds in this context. The *optimal sampling procedure* (Section 7.5.1) does not infringe on probabilistic completeness as random samples are still required to satisfy primitive redundancy.

Similar to Online CSP, the Online MPP employs the same path planning algorithms to solve the MPP. Englot and Hover (2012a) provides a proof of completeness for any MPP algorithm that iteratively solves the MPP using an RRT-based solution. Since the RRT has been proven to be probabilistically complete by LaValle et al. (2001), enables any MPP algorithm utilising an RRT to be probabilistically complete. Since the implementation of the LPP retains an RRT-based solution ensures that the probabilistic completeness holds for path planning in the conducted by the *adaptive sampling-based coverage planner*.

The introduction of a *dummy* configuration into MPP does not affect the completeness and convergence of a solution. Since an RRT is never used to update the costs between the *dummy* configuration and the start and finish locations, the *dummy* configuration cannot be assessed for probabilistic completeness. The *dummy* configuration serves as a virtual path that is only used to ensure the start and finish locations are adjacent in the final solution. As

the *dummy* configuration has no impact on other path solutions and is removed at the conclusion of the MPP solution, results in a final solution of only RRT paths.

The final stage of the Online MPP process merges *preserved path segments* with newly solved sub-plans. Since both the *preserved path segments* and sub-plans have both been proven to be probabilistically complete, the merging of these path segments must create an overall solution that is collectively probabilistically complete. Therefore, even with the presence of *prison cells*, which are compensated for in this work, and functions that segment the tour using ROIs, the sampling and path planning methods still adhere to probabilistic completeness.

7.9 Chapter Summary

In this chapter, the *adaptive sampling-based coverage planner* that uses a ROI to perform a *plan repair strategy* to partially replan segments of the existing tour due to the influences of newly introduced features was proposed. When a map update is supplied from the mapping system, the three-phase planning procedure produces a ROI around the new features to segment the current plan. Regions outside the ROI are preserved whilst regions within the ROI are replanned to cover the new features.

The proposed *adaptive sampling-based coverage planner* still maintains the same sampling and path planning procedures to generate a *redundant roadmap* replan new path segments into an existing tour. By treating each replanning iteration independently ensures that these procedures act as if the planning problem present is equivalent to solving smaller offline plans. Compared to a *full replan strategy*, the *path replan strategy* attempts to reduce the planning effort that, for tank environments, have been proven to be expensive environments to cover.

By developing a new *adaptive sampling-based coverage planner*, this chapter presented a new *set system* (M,Q) that better represents the online planning problem. Throughout this chapter, certain aspects of the *offline sampling-based coverage planner* were amended, and new procedures were introduced to aid the *adaptive sampling-based coverage planner* to work robustly online. These changes included;

1) an *optimal sampling procedure* to enable the sampling procedure to sample optimal positions of the surfaces first before resulting to random sampling,
2) the tracking and handling of occluded primitives during the lifetime of an inspection,

the relaxation of the TCH to create the R-TCH that attempts to *capture trapped configurations* within partially constructed structures,

3) the development of the NNH to aid the R-TCH to capture *trapped configurations* without calculating a motion plan, and
4) provided the LPP with the TCH-PP so undetected *trapped configurations* can be removed during the path planning phase.

Finally, the chapter concluded with a demonstration that the *adaptive sampling-based coverage planner* also adheres to probabilistic completeness, providing the ROI encapsulates all primitives that may be influenced by the new features. The generic nature of the ROI and the preservation of the existing sampling-based coverage and path planning methods enabled it to be possible to preserve this important attribute of the *offline sampling-based coverage planner*.

In the next chapter, the performance of the *adaptive coverage planner* using ROIs is compared against an *adaptive coverage planner* using the *full replan strategy* to determine which of the two replanning strategies is best for the *offline sampling-based coverage planner*. The next chapter will also include a description of the *adaptive coverage planner* using the *full replan strategy* which formulation is heavily based of the terms presented in this chapter.

Chapter 8
Computational Analysis of the Adaptive Sampling-Based Coverage Planners

8.1 Introduction

In the previous chapter, the *adaptive sampling-based coverage planner* using a *plan repair strategy* was proposed in detail. To minimise the computational effort of performing an online tour update, the *plan repair strategy* uses the visibility constraints of the sensor to form a *region of interest* (ROI) around each new primitive of the map update to determine which segments of the current tour are influenced by the new changes. Regions of the environment that are under the influence of new changes are subject to replanning and merged back into the existing tour.

The aim of this chapter is to investigate whether the *plan repair strategy* is a suitable approach to extend the *offline sampling-based coverage planner* as an adaptive online implementation. To determine if this strategy is suitable, a series of experiments were designed to compare the performance of the *plan repair strategy* against an *adaptive sampling-based coverage planner* using a *full replan strategy*. The experiments were designed to answer the following questions:

Question 1) What is the computational impact of segmentation and merging of sub-plans on the planning times?

Question 2) What is the potential drop in tour optimality as the sub-plan approach does not solve the coverage plan globally?

Question 3) How does the *plan repair strategy* scale to the size of the environment and size of the detected features?

Question 4) What is the replanning limit in which the *full replan strategy* outperforms the *plan repair strategy*?

The experimental results obtained in *Chapters 4 to 6* suggest that the *plan repair strategy* should outperform the *full replan strategy*. However, given neither strategy has been applied before to convert the *offline sampling-based coverage planner* to perform adaptively online, in this context, it is unknown whether the additional processes to *segment* and *merge* enable the *plan repair strategy* to successfully outperform the *full replan strategy*.

While the *plan repair strategy* is expected to outperform the *full replan strategy*, there is a limit that exists where the *full replan strategy* will outperform the *plan repair strategy* given the additional computation required to *segment*, *replan* and *merge*. This chapter seeks to determine where this limit exists for the designed *plan repair strategy* given the environments used for testing.

Finally, the analysis of the experiments undertaken in this chapter provide an indication as to whether the *plan repair strategy* is capable of satisfying *Requirement 7* of the *submarine tank inspection planning problem* (STIPP). As there was no prior research available to suggest how either replan strategy will perform within the complex, confined spaces of a submarine tank, it was decided at a one to two-minute replanning time would be a suitable benchmark to achieve. Currently, offline planning results in previous chapters indicate the *full replan strategy* would be in the order of five to seven minutes.

For clarity, to distinguish between the two types of *adaptive sampling-based coverage planners*, the *adaptive coverage planner* executing a *plan repair strategy* is referred to as the AD-P, and the *adaptive coverage planner* using a *full replan strategy* is referred to as the AD-R.

8.2 Experiment Methodology

This section discusses the methodology used to test the two *adaptive coverage planners*. This section covers the following;

1) the proposal of AD-R,
2) the experiments that have been designed to analyse the functionality of AD-P against AD-R,
3) the planning environments used in the experiments,

4) the initial offline tours used to initialise the online adaptive planners,
5) determine the appropriate redundancy to create online *redundancy roadmaps*,
6) contextualise the output of the *adaptive coverage planners* with respect to the concept demonstrator that was intended to be used in the real-world trials, and
7) discuss how the *additional termination conditions* (*Chapter 5*) and *hybrid-heuristic* (*Chapter 6*) are integrated within each of the *adaptive coverage planners*.

8.2.1 Adaptive Coverage Planning Using a Full Replan Strategy

To enable an unbiased comparison between *adaptive coverage planners*, AD-R is also designed to execute using the same ROI Validation, Online CSP and Online MPP procedures as AD-P. However, as AD-R is not required to preserve any configurations not achieved by the robot during execution, there are a few notable changes to the planning process.

Figure 8-1 presents the AD-R architecture using the new *set system* (M, Q) representation as defined in Section 7.2.1. To ensure inspected areas are not replanned again, the ROI Validation phase is only used to verify which configurations q_j, of the current tour T, have been achieved by the robot Q^{VALID}. All remaining configurations and their associated coverage are removed $Q^{COMPROMISED}$, to allow unhindered replanning. The primitives p_i of the original environment P, that become unobserved due to this validation process along with all the new primitives P^*, of the map update M, are supplied together U, to the Online CSP. The Online CSP performs as intended. A *redundant roadmap* Q^{NEW}, is created and pruned to create a minimal covering set S^*, by solving the *set cover problem* (SCP) that observes U.

As the Online MPP assumes the current position and the final position are always valid positions, the Online MPP for a *full replan* creates a *single unresolved path segment* l^U, to reorder S^*. The *unresolved path segment* is solved as a *travelling salesman sub-problem* (Section 6.6.1) using the LPP, with the resultant *ordered path segment* appended to Q^{VALID} to create a new tour T^*. The AD-R continues this process until the inspection plan is complete.

As the entirety of the coverage plan is replanned beyond the current position of the robot, the *full replan strategy* still adheres to the *probabilistic completeness* (Englot and Hover, 2012a; Section 7.8). Unlike AD-P, which requires all primitives and configurations influenced by changes to be correctly identified, AD-R requires any primitives that have not

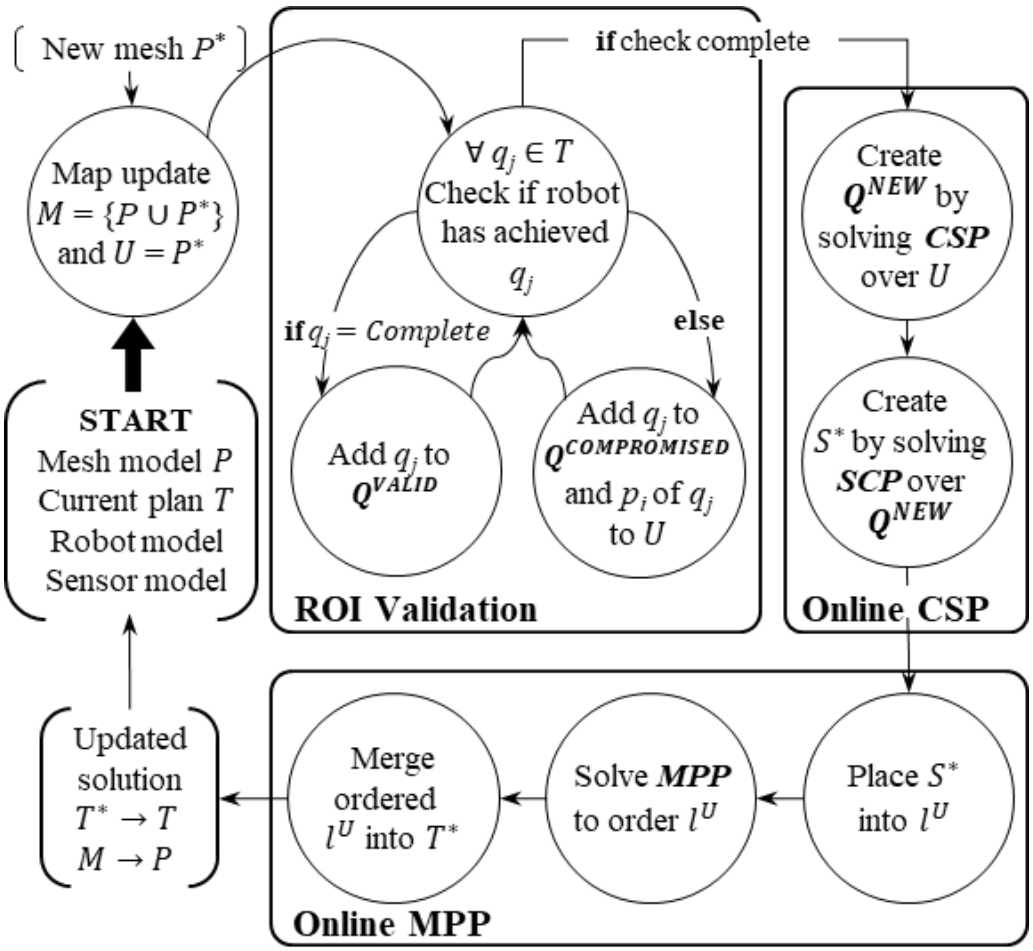


Figure 8-1: Flowchart representation of *adaptive sampling-based coverage planner* using the *full replan strategy*.

been captured by the robot before this event, to be covered again. Providing configurations trapped in *prison cells* are removed, *probabilistic completeness* is still maintained.

8.2.2 Experiment Design

Three simulated experiments, listed below, were designed to analyse the functionality of AD-P to answer the questions listed in Section 8.1. The experiments conducted in this chapter are:

Experiment 1: Size and scalability of AD-P within a controlled environment with full map updates (Section 8.3).

Experiment 2: Adaptive planning within the representative tank environments with full map updates (Section 8.4).

Experiment 3: Coverage planning with partial map updates in the controlled and representative tank environments (Section 8.5).

Experiments 1 and *2* that plan with full map updates, do not simulate the execution of the

coverage plan or interact with a mapping system. Map updates are delivered in full at the beginning of the trial to simulate the maximum replanning effort over one planning iteration. These experiments demonstrate the principles and intrinsic behaviours of AD-P and AD-R before being applied to perform iteratively over partial map updates in *Experiment 3*, where the *adaptive coverage planners* are coupled with a mapping system (Section 8.2.4) to simulate the coverage plans. This experiment will provide an indication whether AD-P and AD-R can replan under the one to two-minute time limit as stipulated by the STIPP criteria.

The experiments in this chapter compare the performance of both replanning strategies over controlled environments and representative tank environments. Controlled environments are again used to analyse the intrinsic behaviour of the new *adaptive coverage planners* and the representative tank environments will provide validation to the previous results. Complexity is introduced into these planning problems by incrementally adding more features into the environment. The more features that are introduced, the more replanning that is required. While *Assumption 2* assumes that in a practical scenario only small changes are expected to occur inside the submarine tanks, 30% maximum, these experiments will introduce enough change within the environment to influence over 50% of the original tours. This will help determine if AD-P is capable of replanning faster at higher replan efforts. More details about the environments used and the introduced changes are discussed in Section 8.2.3.

For consistency across all experiments in this thesis, the simplified robotic platform and visibility constraints, as discussed in Section 3.6, are again used to minimise the number of extrinsic influences on the planning procedures. As the robot platform has been simplified, the tours that are produced in these experiments are only representative of what a multi-legged robot may perform. For robotic platforms that share similar 6-DOF holonomic constraints, the results are transferable.

All experiments in this chapter were conducted on the same 64-bit Intel i7 920 CPU, 8 core, 6GB RAM machine running Ubuntu 16.04 LTS as in *Chapters 4-6*. Analysis of the data was performed with MATLAB 2018b. Appendix A contains the list of open source packages that were used to create both AD-P and AD-R.

8.2.3 Planning Environments

Expanding upon 2*x2m* and *6x6m* used in previous chapters, seven controllable box environments, increasing in size from 2.0 x 2.0 x 0.2m (*2x2m*) to 8.0 x 8.0 x0.2m (*8x8m*), are used for *Experiments 1* and *3* (Figure 8-2). Complexity inside these environments is

controlled by introducing the same single cylindrical feature into the environment, at equidistant locations from the sides of the environment and from each other. Figure 8-2c shows the order and placement locations of up to nine cylindrical objects that were incrementally introduced in full for *Experiment 1*. The placement of the cylindrical features ensured that 100% coverage was obtained. Even in the smallest environment, *2x2m*, the new features were placed far enough apart to avoid primitive occlusion between objects.

By introducing the same feature each time, it was expected that for each replanning update, the planning times for AD-P should scale approximately linearly. Introducing the same feature allowed for more deterministic behaviours to be observed by the *adaptive coverage planner*. The influence of the same feature across the trials of the same planning problem should lead to;

1) the same number of configurations being inside ROI,
2) the same number of path segments being preserved,
3) approximately the same sized covering sets being produced for each feature, given that the same number of primitives are being added to the planning problem,
4) approximately the same number of path evaluations to solve multiple MPPs, which should therefore lead to,
5) approximately the same planning times being achieved for each trial for each planning problem.

By restricting the replanning effort by introducing the same features, ensures that the only variability that should be observed across the trials is that of the random nature of the sampling and path planning procedures. This enables a thorough analysis of AD-P to be performed in order to understand the intrinsic behaviour of the AD-P, without the influence of irregular partial map updates, which would otherwise make it difficult to compare against AD-R.

It was expected that in the smallest planning environment with the maximum amount of change (*2x2m_9*), AD-P would be required to replan upwards of 85% of the existing tour. This would present as the toughest problem for AD-P to solve and it was expected that AD-P both planning times and tour quality would be less than AD-R due to solving the replanning problem locally. The easiest replanning problem, with respect to how much of the tour should be preserved, was expected to be *8x8m_1*. Between these two extremes, the immediate planning environments would demonstrate the limits of operation for AD-P,

highlighting under controlled conditions, where the expected degradation occurs with the *plan repair strategy*.

Representative tank environments *Tank and Tank-P4* (Figure 8-3) were again used to analyse how the *adaptive coverage planners* perform in the target environment. To incrementally increase complexity of these environments, the four pipes that exist upon the sides of *Tank-P4* were incrementally introduced one at a time (*Tank-P1*, *Tank-P2* and *Tank-P3*) until *Tank-P4* was created. The introduction of each pipe increased the replanning effort, with an expectation that *Tank-P4* would influence upwards of 40% to 50% of the existing tour, which would be a greater influence than what was expected in the real-world (*Assumption 2*). The specifications of the mesh density and mesh resolution for each planning environment is detailed in Table 8-1.

For conciseness, *House* and *House-W* are excluded from the experiments. Results in *Chapter 6* demonstrated the impact that a small change in geometry had on the *lazy point-to-point planner* (LPP). The *hybrid-heuristic* was developed to overcome the underestimation and the results indicated the *hybrid-heuristic* was sufficient to provide the coverage planner with a better approximation of the connectivity. Given *House-W* only introduces one small change, the results from the controlled and representative tank environments were deemed suitable enough to analyse AD-P and AD-R.

8.2.4 Simulated Map Updates and Robot Motion

For *Experiments 1* and *2*, no mapping system was required to perform the experiment. Therefore, only a single replanning iteration was performed. To replicate a map update, both AD-P and AD-R were initialised with a complete map of the environment. To delineate between primitives of the original map and the primitives of the new features, a threshold (m_{index}) was provided (*Assumption 3*; Sections 3.6.1 and 7.4.1).

For *Experiment 3*, a mock-mapping system was developed based on the assumptions listed in Section 3.6.1, to deliver map updates to each *adaptive coverage planner*. In practice, the mapping system to be implemented on the physical platform is described in further detail in Pivetta et al. (2017). However, for this experiment, the simulated robot platform was assigned with a one metre sensing capability to emulate the lidar.

To detect primitives within this sensing range, a mock map system would perform a *nearest neighbours' search* each time the robot was instructed to move along the inspection plan.

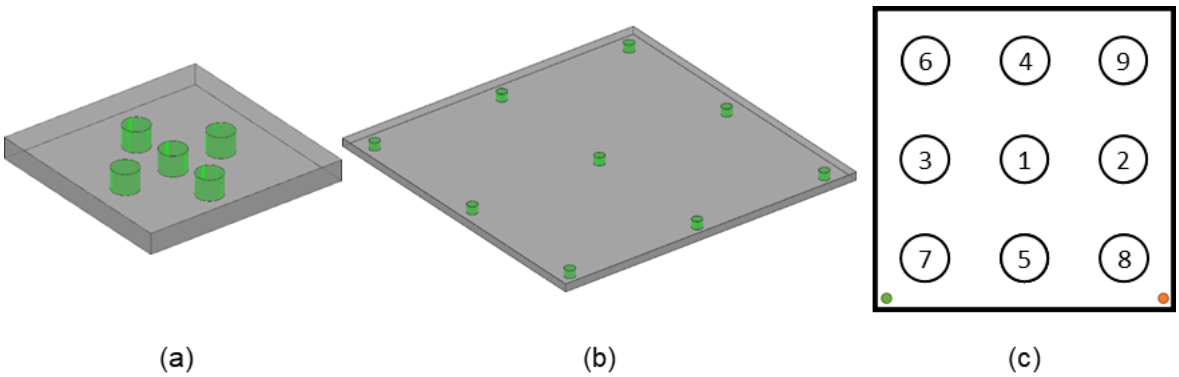


Figure 8-2: Controlled simulation environments and update order. (a) The smallest *2x2m* environment containing five changes (green). (b) The largest planning environment *8x8m* with the maximum nine changes. (c) The order in which each update will appear in the experiments. The green and red dots indicate the designated start and end positions for every plan respectively.

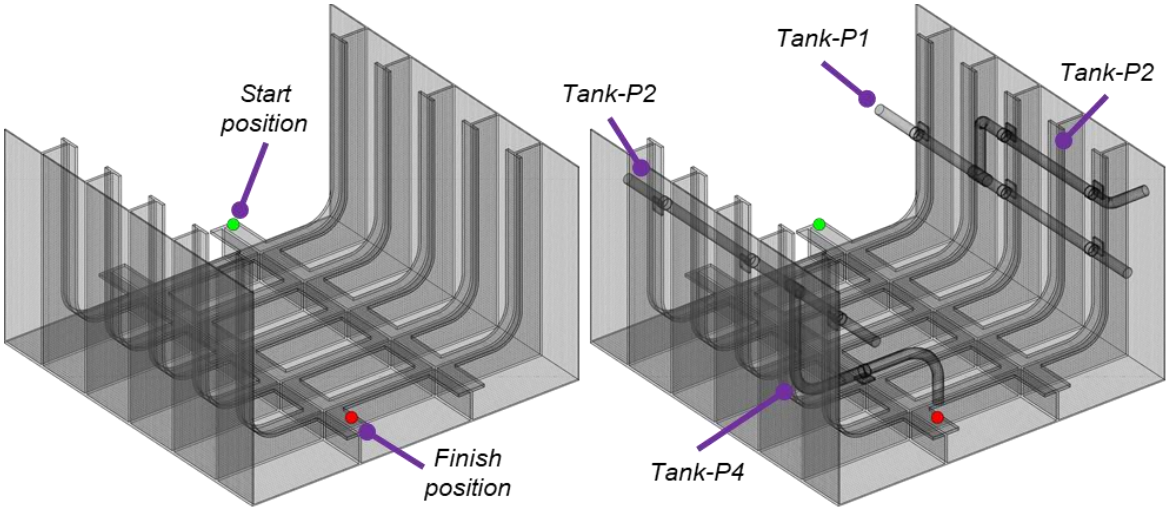


Figure 8-3: *Tank* and the additional pipe network to construct *Tank-P4*.

Table 8-1: Resolution and primitive counts for controlled and representative environments.

Controlled Environments			*Representative Tank Environment*		
Model	*Resolution (mm)*	*Primitives*	*Model*	*Resolution (mm)*	*Primitives*
2x2m	*13.1*	*58,058*	*Tank*	*18.0*	*273,382*
3x3m	*13.5*	*108,894*	*Tank-P1*	*12.5*	*+17,732*
4x4m	*13.8*	*200,448*	*Tank-P2*	*8.5*	*+31,772*
5x5m	*14.0*	*293,113*	*Tank-P3*	*12.5*	*+17,732*
6x6m	*14.0*	*410,432*	*Tank-P4*	*7.0*	*+31,336*
7x7m	*14.0*	*549,450*	*Total*	*-*	*371,954*
8x8m	*15.8**	*586,569*			
Cylinder	*15.1*	*4,550*			

** Different resolution used to reduce file size*

These new primitives were appended to the existing mesh representation of the environment to create a new map update (M) in the form of a STL (Stereolithography) file. The STL file along with m_{index} were provided to each adaptive planner for each map update.

To avoid receiving a map update for each detected primitive, a threshold of 100 primitives was used to trigger a map update. If the robot arrived at a viewing location and primitives have been detected that are under the threshold, a map update was supplied and updated before continuing. The 100 primitive threshold was only in place when the robot travelled between viewing locations.

Primitives that become occluded due to *prison cell* (Section 7.4.2) are not removed from the mapping updates. These primitives are recorded in the *unobservable primitive list* and maintained throughout the lifetime of the inspection as discussed in Section 7.5.2. This allows primitive indices of *a priori* mesh to remain consistent during the inspection and allows the assumption of only new primitives entering the planning problem to be adhered too.

8.2.5 Initial Offline Tours

To remain consistent with the results from previous chapters, the same visibility constraints are used to initialise the robot model. The viewing range remains between 100-to-270mm with a *field of view* (FOV) of ±45° with the robot end-effector size set to 50mm. Using these constraints, the initial offline tours for each environment were created from randomly sampled redundancy-ten roadmaps generated by the recreated *offline sampling-based coverage planner* (Section 4.2). The offline tours are solved between designated start and finishing locations using the *Travelling Salesman Sub-problem* described in Section 7.6.1 (Figure 8-2c). Details of the random offline inspection plans for each environment can be found in Table 8-2.

Experiments conducted within the controlled planning environments are also initialised with a pre-determined raster plan. Initialising the *adaptive coverage planners* with different tour types demonstrates the ability of the AD-P to replan any type of offline coverage plan, but aid in the analysis to determine under controlled conditions if different behaviours are exhibited with a different initial tour. For consistency with generated random plans, the raster plans start and finish at the same locations as the random plans.

Table 8-2: Offline random and raster coverage plans to initialise the *adaptive coverage planners.*

Model	*Random Offline Plan Attributes*			*Raster Planning Attributes*		
	Configs	*Tour Length (m)*	*Tour time (mins)**	*Configs*	*Tour Length (m)*	*Tour time (mins)**
2x2m	*156*	*20.34*	*8.59*	*142*	*21.38*	*8.30*
3x3m	*303*	*42.79*	*17.23*	*254*	*44.36*	*15.86*
4x4m	*491*	*72.75*	*28.49*	*398*	*75.34*	*25.82*
5x5m	*719*	*109.71*	*42.25*	*574*	*114.31*	*38.19*
6x6m	*999*	*155.70*	*59.25*	*782*	*161.29*	*52.95*
7x7m	*1,356*	*210.60*	*80.30*	*1,022*	*216.27*	*70.11*
8x8m	*1,738*	*274.89*	*103.75*	*1,294*	*297.25*	*92.68*
Tank	*1,319*	*139.36*	*67.19*	-	-	-

Configs - Configurations *CSP - Coverage Sampling Problem* *MPP - Multi-goal Planning Problem*

Pre-determined raster plans were calculated to a minimum of 99.5% coverage. Given the visibility constraints, complete coverage of the corners was not achievable by the standard raster. These missing primitives were initialised as *unobservable primitives* (Section 7.5.2) at the beginning of a planning iteration and neither *adaptive coverage planner* was required to replan the missing coverage. *Adaptive coverage planners* initialised with a raster plan are denoted with '(R)', such that AD-P is AD-P(R) and AD-R is AD-R(R). The standard notation AD-P and AD-R will represent the initialisation of each *adaptive coverage planner* with a random plan.

The *optimal sampling procedure* (OSP) presented in Section 7.5.1 was not used to create the initial offline tours to ensure random variability between configurations was present. If the OSP were to be used in the controlled environments, it would produce a uniform spread of configurations throughout the environment making the plans similar to the predetermined raster plans. The inclusion of the OSP into the online planning problem is discussed in more detail in Section 8.2.7.

8.2.6 Contextualised Planning Outcome: Tour Time and Relative Computational Effort

To contextualise the results of the simulated experiments to the target robotic platform (Section 1.2.2), a cost metric denoted as *tour time* was calculated from the final tours to determine how long the respective tours take to execute. *Tour time* was calculated using the two metrics of the inspection plan, the overall length and the number of configurations that comprise the tour. Giving the robot a speed of 0.1m/s to move between locations and two

seconds at each viewing location to obtain the coverage, the tour length and the number of viewing configurations create the cost metric *tour time* that approximates, in seconds, how long the inspection will take to execute (8-1).

$$Tour\ time = (2.0s * \#Configurations) + (Tour\ Length\ /\ 0.1m/s)\ (\text{s}) \qquad (8\text{-}1)$$

Table 8-2 shows the *tour times* for offline raster and random tours. While the tour lengths are longer for the raster plans, the fewer number of configurations these tours possess create *tour times* that will execute in a shorter time than their respective random tours.

In Section 3.6.1, *Assumption 7* states that replanning was assumed to occur concurrently as the robot performs the inspection. Therefore, given this assumption, the overall planning time (OPT) of each of the *adaptive coverage planners* are not included in the *tour time* calculation. Removing the OPT from the *tour time* calculation allows a comparison to occur between the computational effort exhibited by each *adaptive coverage planner*. It was expected that AD-P will be faster than AD-R. However due to AD-P replanning locally, a degradation in tour quality was expected. Essentially, there was a trade-off between the time it takes to calculate the tour and the effect the tour degradation has on time to complete the inspection plan given the constraints provided. This trade-off creates a *relative computational effort* (RCE) that seeks to answer the primary questions of this chapter.

1) How much longer is the AD-P *tour time* compared to the *tour time* of AD-R?
2) How much longer is the OPT of AD-P compared to OPT of AD-R?

Equation 8-2 presents the mathematical form of the RCE defined by the two questions.

$$RCE = \left(\frac{AD{-}P\ Tour\ Time}{AD{-}R\ Tour\ Time}\right) / \left(\frac{AD{-}P\ OPT}{AD{-}R\ OPT}\right) \qquad (8\text{-}2)$$

The RCE aids in the comparison between AD-P and AD-R because it was expected that tour degradation may occur however with the computational advantages AD-P should present in the smaller replanning situations, the extra tour length may present to be the better option given the computational times of AD-R.

8.2.7 The Selection of a Suitable Redundancy for Online Planning Using the Optimal Sampling Procedure to Create the Redundant Roadmaps

In Section 7.5.1, the OSP was proposed to introduce the option to sample the preferential *optimal* viewing location first before reverting to random samples to complete the *redundant roadmap*. The aim of introducing the OSP was to;

1) minimise the number of random configurations to cover flat geometries,
2) produce more regular higher-quality observations in areas of lower geometric complexity, and
3) maintain a level of consistency between configurations that would assist with the motion of the robot with smoother transitions between configurations

To test the applicability of the OSP as the new sampling procedure to solve the *coverage sampling problem* (CSP), OSP was compared against *redundant roadmaps* generated using a fully random sampling procedure. The smallest and largest planning environment of each of the controlled and representative series, with respect to the number of primitives in the planning environment, *2x2m*, *8x8m_9*, *Tank* and *Tank-P4,* were used to determine an appropriate redundancy level for online planning (Table 8-1). The *optimal* viewing distance was set at 170mm (Table 3-1) with random configurations drawn to the simplified constraints discussed that made the initial tours in (Section 8.2.5). To determine the effectiveness of the OSP, each *redundant roadmap* was trialled 20 times over increasingly larger *redundant roadmap* sizes. Figures 8-4 to 8-7 present the results of these trials.

For smaller redundancies sizes (1 to 5), the covering set sizes produced by the OSP for the controlled planning environments (*2x2m* and *8x8m*) were 6.4% to 16.9% smaller than randomly drawn *redundant roadmaps*. The representative tank environments were 2.5% to 9.2% smaller than the respective randomly sampled covering sets. Over these smaller redundancies, majority of the samples in the OSP generated roadmaps are optimal. In the controlled environments, optimal samples dominate the *redundant roadmaps* upwards of 84% to 99%. Complete coverage within these environments could not be achieved due to the collision of optimally drawn positions in the corners. Therefore, random samples were required to fill the remaining coverage. In more complex geometries, random sampling had to occur, therefore, the use of optimal samples decreased to 64% to 78% for *Tank* and *Tank-P4*. Overall, the OSP has provided covering set sizes of fewer configurations in comparison to the random sampling procedure at lower redundancies.

For larger redundancies greater than 10, the covering sets produced by random sampling were smaller than the covering set sizes generated by the OSP. Coverage set sizes generated by random sampling over all environments were 10% to 20% smaller than OSP sets. The disparity between the two sampling methods at higher levels of redundancy is a consequence of the OSP and the selection of criteria of the *set cover problem* (SCP).

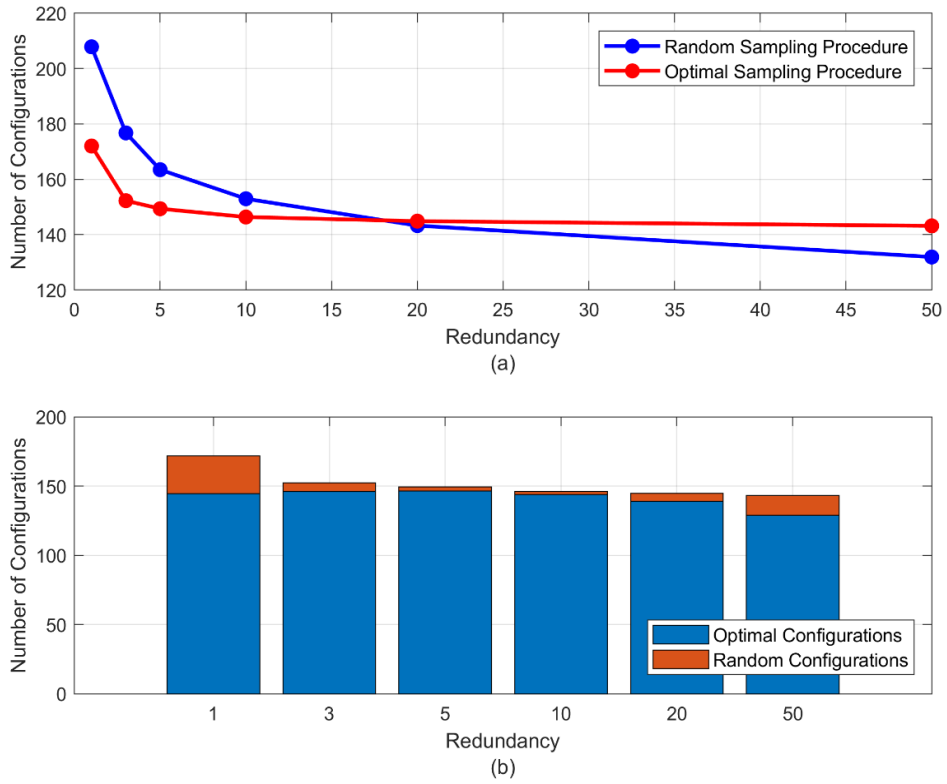


Figure 8-4: *Optimal Sampling Procedure* (OSP) over *2x2m*. (a) Covering set sizes of varying redundancies using the OSP against a fully random roadmap construction. (b) Number of *optimal* and random samples taken in the final tour for a *redundant roadmap* constructed by the OSP.

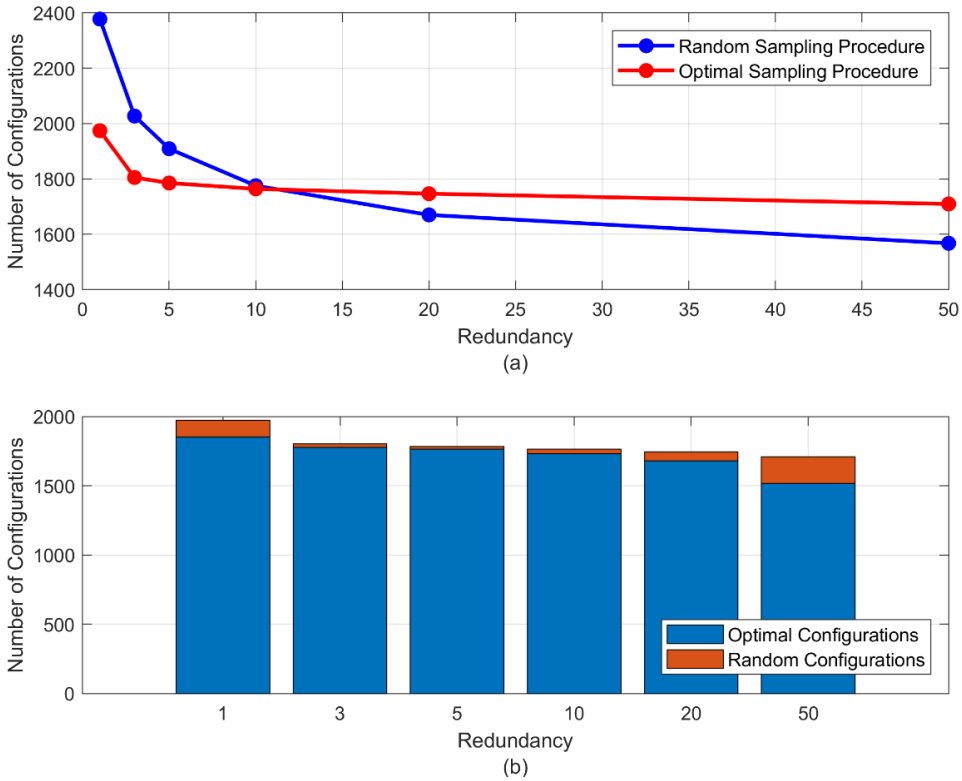


Figure 8-5: *Optimal Sampling Procedure* (OSP) over *8x8m_9*. (a) Covering set sizes of varying redundancies using the OSP against a fully random roadmap construction. (b) Number of *optimal* and random samples taken in the final tour for a *redundant roadmap* constructed by the OSP.

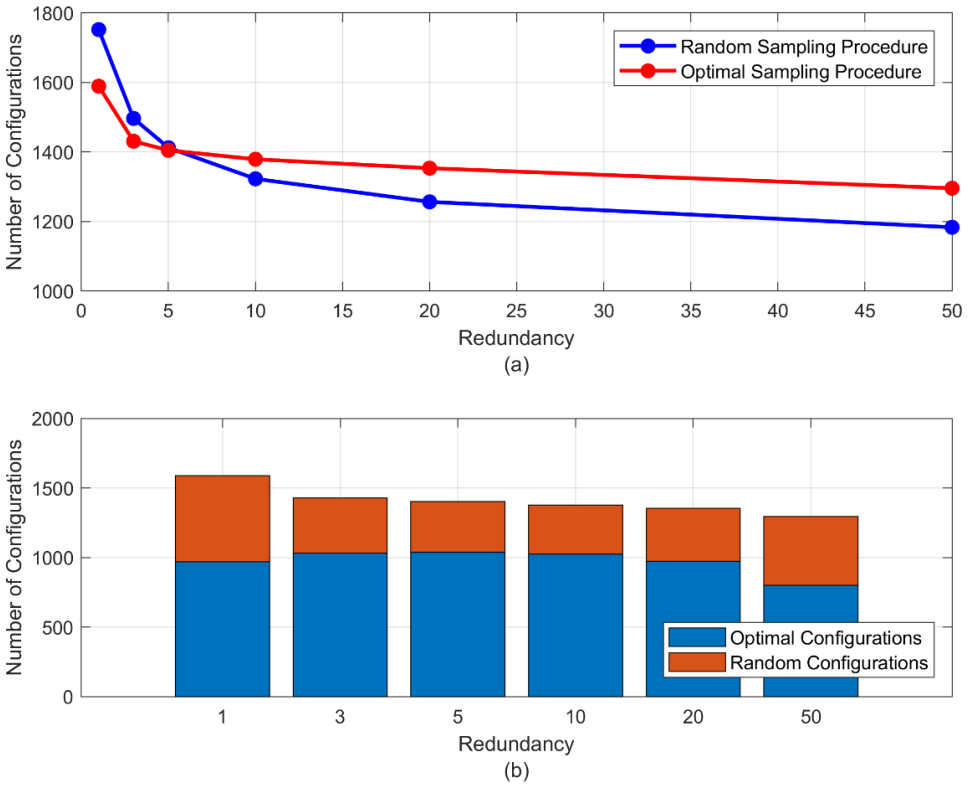


Figure 8-6: *Optimal Sampling Procedure* (OSP) over *Tank*. (a) Covering set sizes of varying redundancies using the OSP against a fully random roadmap construction. (b) Number of *optimal* and random samples taken in the final tour for a *redundant roadmap* constructed by the OSP.

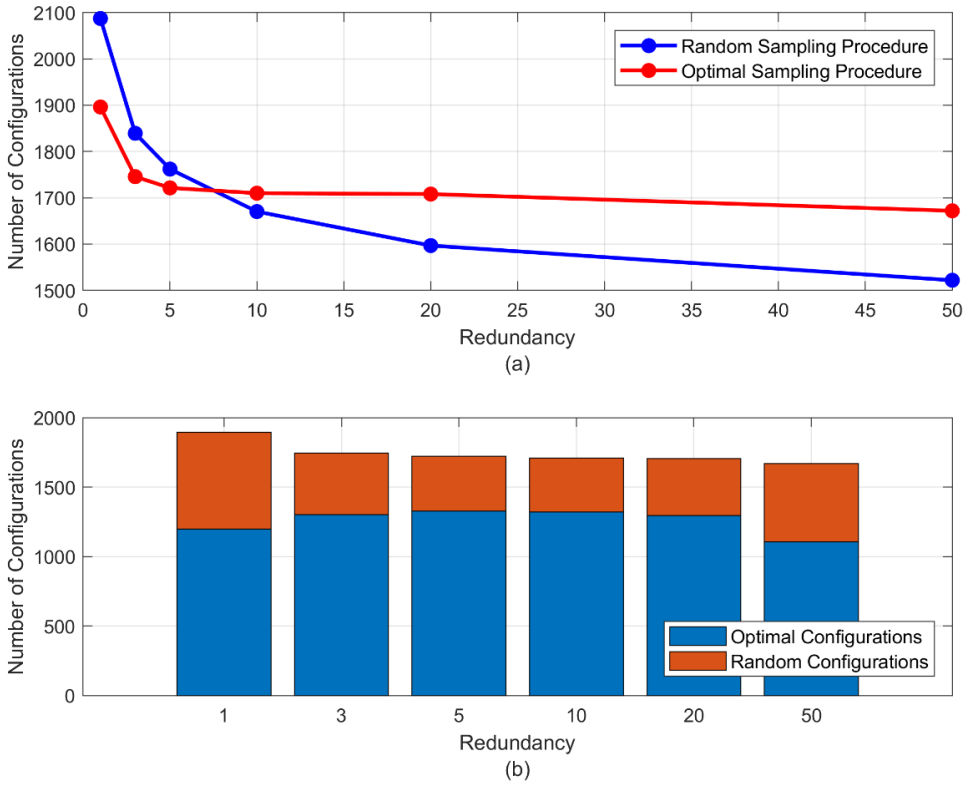


Figure 8-7: *Optimal Sampling Procedure* (OSP) over *Tank-P4*. (a) Covering set sizes of varying redundancies using the OSP against a fully random roadmap construction. (b) Number of *optimal* and random samples taken in the final tour for a *redundant roadmap* constructed by the OSP.

The SCP selects configurations based on quantity not quality. While the *optimal* position presents as the best position to observe the surface, it is not necessary. The parameters used for FOD_{max} was 220mm and the *optimal* position being placed at 170mm. Therefore, there is a small range above the *optimal* position that can be sampled that may result in a configuration being able to observe more primitives than the *optimal* (Figure 8-8). Consequently, the randomly drawn configurations above the optimal will be added to the set cover.

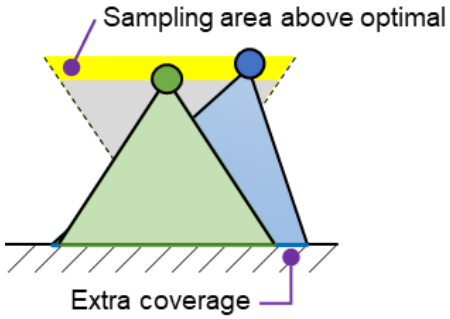


Figure 8-8: Configurations sampled above the *optimal* viewing location viewing the same primitive are likely to be taken in the set cover due to the extra primitives they might observe.

Over flat geometries, it is highly likely redundancy will be filled by *optimal* configurations, as surrounding *optimal* configurations also observe locally grouped primitives, before the same primitive is likely to be sampled again. Therefore, fewer random samples have the opportunity to be sampled above the *optimal* position. However, the random sampling procedure has more opportunity to sample from the entire neighbourhood around the primitive, increasing the chances configurations could be drawn above the optimal threshold.

Given the results from this experiment, it has shown that the OSP is not suitable for *redundancy-ten roadmaps* or higher if minimising the number of configurations in the covering set is the main objective. To best utilise the OSP for online planning, *redundant roadmaps* should be constructed from smaller redundancies given that the OSP did produce smaller covering set sizes compared to the random sampling procedure. Therefore, given the outcomes of this experiment, a *redundancy-five roadmap* was chosen to efficiently utilise the OSP in the online experiments. While it would be preferable to use a lower redundancy to allow AD-P to perform efficiently, to ensure an even comparison between AD-P and AD-R, a *redundancy-five roadmap* was chosen to enable AD-R to produce tours containing similar set size to AD-P.

8.2.8 Inclusion of the Additional Termination Conditions and Hybrid-heuristic into the Adaptive Coverage Planners

The *additional termination conditions* and *hybrid-heuristic* developed in *Chapters 5* and *6* respectively are utilised in these experiments. Regardless of the size of the replan effort the *additional termination conditions* will have a positive impact of minimising the influence of the *state of minimal improvement*. Given the results in *Chapter 6*, the *hybrid-heuristic* would only be applied to the representative tank environments. The calculation of the *hybrid-heuristic* for the controlled environments is not expected to improve upon either the MPP time or tour quality, as the MPPs would be primarily solved using the *Euclidean assumption*.

As a *full replan strategy* in *Experiment 1* and *2* is equivalent to generating an offline plan, there is an expectation that the *hybrid heuristic* would have a positive influence on Online MPP times. However, it is unknown what influence the *hybrid-heuristic* would have on aiding the *plan repair strategy* as it depends upon the size and influence of the detected changes. If the replan effort is small, the computation of the *hybrid-heuristic* may impact planning times despite the potential of solving the MPP more efficiently than using the *Euclidean assumption*. Otherwise, if the replan effort becomes greater, the positive impact of the *hybrid-heuristic* will increase with the increasing size of the replan effort.

Figure 8-9 highlights how the *hybrid-heuristic* is incorporated into both the AD-P and AD-R. For the AD-R, the *hybrid-heuristic* is implemented in the same way as the offline MPP (Figure 8-9a). The covering set, as generated by the Online CSP, along with the current and finishing location, are supplied to the LPP to construct an initial estimation of the connectivity. As for the AD-P, to avoid solving the *hybrid-heuristic* for each sub-MPP, it is only solved once over the entire covering set and ROI gate pairs. To initialise the LPP for each sub-MPP, sub-adjacency matrices were created from the aforementioned solution (Figure 8-9b).

For clarity, the *adaptive coverage planners* that utilise the *hybrid-heuristic* are denoted with (HY), e.g. AD-P(HY) and AD-R(HY). The standard notation AD-P and AD-R will represent the initialisation of the LPP with the *Euclidean assumption*. *Tank*, and the *Tank* variants, are voxelised to a 15mm resolution, consistent with the resolutions used in *Chapter 6*.

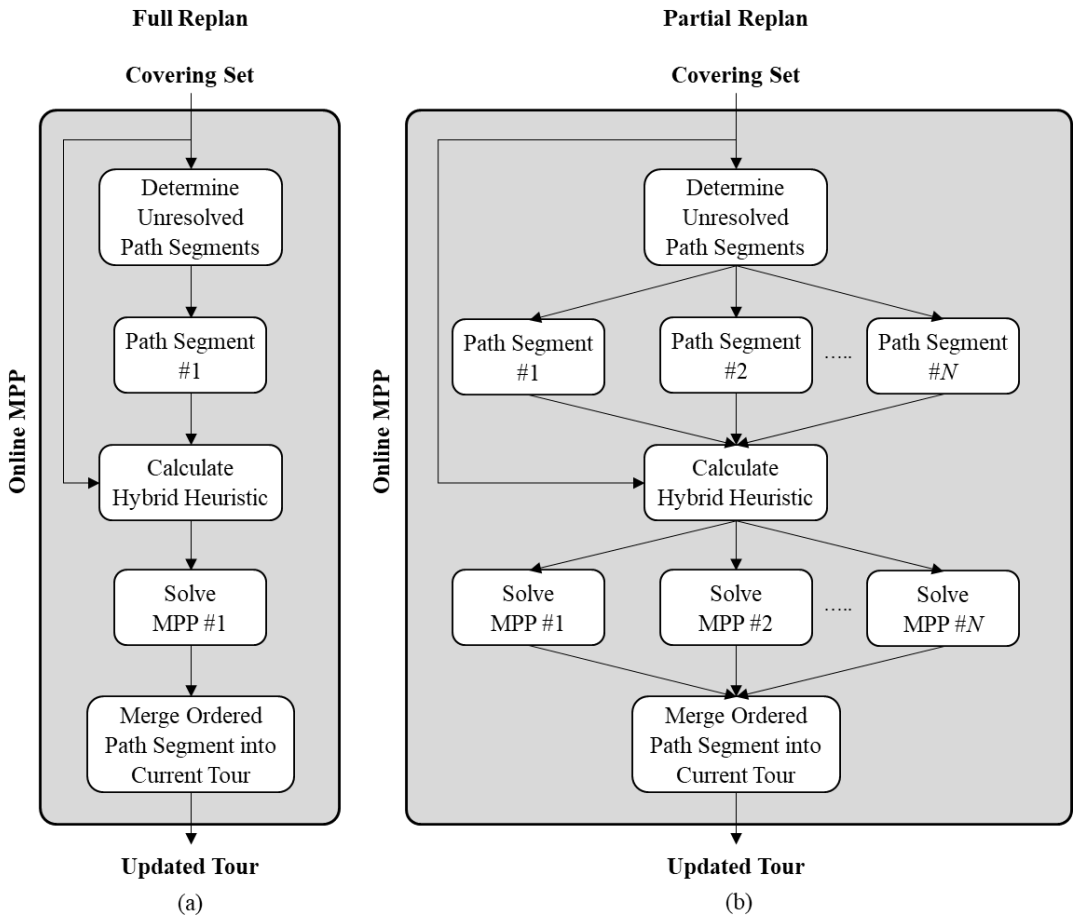


Figure 8-9: Implementation of the *hybrid-heuristic* for each *adaptive coverage planner*.

8.3 Experiment 1: Size and Scalability of AD-P within a Controlled Environment with Full Map Updates

Experiment 1 was designed to determine the sensitivity of the AD-P to changes in the environment. The AD-P is designed such that the replan effort focusses exclusively on the areas or segments of an existing tour where changes have occurred. This experiment used the controlled environments prescribed in Section 8.2.3 to systemically control the replanning effort of AD-P by incrementally introducing a *singular cylindrical feature*. By introducing the same feature inside increasingly larger environments, the AD-P was expected to exhibit a similar behaviour in;

1) overall planning times,
2) the number of new configurations introduced to cover new features regardless of the size of the environment, and
3) the number of path evaluations required to solve all MPPs.

A *single replanning iteration* is used to solve each planning problem to represent the full replanning effort required to perform a tour update. From this controlled and systematic

approach, the following questions were investigated;

1) How much replanning of the tour would need to occur to allow AD-P to be just as expensive as replanning using AD-R?
2) How much tour degradation of the resultant AD-P is acceptable before it is beneficial to use AD-R?

To answer these questions, *Experiment 1* tested the following *adaptive coverage planners*, AD-P(R), AD-P and AD-R.

Given the simplicity of the planning environments, the *hybrid-heuristic* was not employed in this experiment. Each planning problem was trialled 50 times by each *adaptive coverage planner*. In total, 9,450 trials were conducted across all planning environments and *adaptive coverage planners*. For comparisons between trials, an independent-samples t-test ($\alpha = 0.01$) was conducted to investigate whether a statistically (p) and practically (d) significant result was present within the trials of the *adaptive coverage planners*. Results of these tests represent the collective result of all planning problems and planning attributes unless specified otherwise. For clarity, the analysis that follows is separated into sub-headings to discuss the main findings of the experiment.

8.3.1 Computational Observations and Results

Analysis of the Replanning Effort Required to Perform a Planning Update

Table 8-3 provides a summary of the statistical analysis of the results performed over all the trials. As expected, small planning environments (*2x2m* to *4x4m*) required the most replanning effort, with replanning effort referring to the percentage of the final tour that was replanned. Introducing more features into the confined space of these environments, resulted in more ROI overlap and therefore increased the influence that each new feature had on the initial tour. As a result, more replanning was required to resolve the plan.

The largest replanning effort for AD-P and AD-P(R) was *2x2m_9* at 84.9% and 94.6% respectively. As the environments grew larger in size, with fewer features, the total replan effort steadily decreased to 1.3% and 2.0% (*8x8m_1*) for AD-P and AD-P(R), respectively. Despite replanning efforts of up to 94.6% of the final tour occurring, no solution calculated by either AD-P or AD-P(R) took longer to compute than any solution calculated by AD-R for the respective planning problem.

Table 8-3: Statistical analysis of AD-P(R), AD-P and AD-R.

Model / Planner / # Features			Planning Time (s)		ROI Validation (s)		Online CSP (s)		Online MPP (s)		Tour Length (m)		Configurations				Planning Iterations#		Path Evaluations#		Tour Replanned (%)*	Tour time (mins)^	RCE
			$\bar{x}$	SD	$\bar{x}$	SD	$\bar{x}$	SD	$\bar{x}$	SD	$\bar{x}$	SD	$\bar{x}$	SD	New	Removed	$\bar{x}$	SD	$\bar{x}$	SD			
2x2m	AD-P(R)	1	0.37	0.01	0.16	0.002	0.21	0.01	0.005	0.002	22.22	0.20	147.46	0.61	9.46	4.00	11.40	2.28	31.38	3.45	17.38	8.62	-
		9	2.84	0.09	0.34	0.003	2.26	0.05	0.21	0.07	22.15	0.31	186.82	1.95	70.94	26.12	10.68	2.26	198.58	11.42	94.62	9.92	-
	AD-P	1	0.38	0.01	0.16	0.002	0.21	0.01	0.01	0.002	21.21	0.31	162.00	0.76	8.00	2.00	9.50	2.11	27.08	5.30	13.66	8.94	5.07
		9	2.84	0.07	0.32	0.004	2.35	0.06	0.16	0.04	30.29	0.46	206.64	2.19	74.58	23.94	79.84	8.18	259.54	13.08	84.92	11.94	1.68
	AD-R	1	1.92	0.10	0.01	0.001	1.68	0.06	0.20	0.08	21.82	0.33	157.40	3.66	155.40	154.00	3.50	1.43	164.68	13.08	100.00	8.88	-
		9	4.22	0.24	0.02	0.002	3.69	0.09	0.48	0.23	23.73	0.29	198.10	3.33	196.10	154.00	4.54	2.08	223.44	24.23	100.00	10.56	-
3x3m	AD-P(R)	1	0.50	0.01	0.27	0.006	0.23	0.01	0.004	0.001	45.10	0.19	260.52	0.91	10.52	4.00	9.36	1.48	24.88	2.54	8.68	16.20	-
		9	2.86	0.08	0.46	0.005	2.34	0.08	0.05	0.008	52.54	0.65	307.20	2.03	78.40	25.20	81.40	6.41	251.26	10.60	72.57	19.00	-
	AD-P	1	0.54	0.01	0.29	0.003	0.23	0.01	0.01	0.001	43.70	0.10	309.50	0.89	10.40	3.90	8.98	1.13	30.72	2.46	9.24	17.60	7.50
		9	3.05	0.08	0.50	0.004	2.45	0.07	0.08	0.03	50.79	0.52	368.96	3.16	90.48	24.52	98.58	11.53	313.72	16.54	72.28	20.76	2.40
	AD-R	1	4.03	0.44	0.03	0.001	3.35	0.17	0.61	0.38	45.32	0.44	303.06	4.45	301.06	301.00	4.74	2.88	330.14	22.44	100.00	17.66	-
		9	6.83	0.74	0.04	0.001	5.52	0.24	1.23	0.66	47.77	0.43	343.42	4.19	341.42	301.00	6.94	3.69	405.96	38.68	100.00	19.41	-
4x4m	AD-P(R)	1	0.77	0.01	0.49	0.005	0.27	0.01	0.004	0.001	76.14	0.17	405.10	1.02	11.10	4.00	9.80	1.60	26.00	3.09	5.72	26.19	-
		9	3.51	0.08	0.67	0.006	2.78	0.08	0.05	0.005	82.22	0.61	462.40	2.80	96.40	32.00	77.20	4.18	248.58	8.01	49.72	29.12	-
	AD-P	1	0.78	0.01	0.52	0.005	0.25	0.01	0.005	0.001	73.49	0.30	496.46	0.99	7.46	2.00	7.96	0.78	23.98	1.93	4.53	28.80	11.55
		9	3.47	0.08	0.77	0.006	2.60	0.07	0.08	0.018	80.71	0.68	543.40	2.13	72.22	19.82	83.84	8.31	285.86	13.93	45.43	31.57	3.72
	AD-R	1	9.10	1.10	0.05	0.001	7.01	0.39	1.98	1.09	76.98	0.43	491.46	5.92	489.46	489.00	8.44	4.63	574.70	42.08	100.00	29.21	-
		9	12.69	1.73	0.06	0.005	9.71	0.46	2.84	1.61	79.42	0.49	533.60	4.23	531.60	489.00	10.28	5.74	653.28	48.74	100.00	31.02	-
5x5m	AD-P(R)	1	1.02	0.02	0.71	0.010	0.30	0.02	0.005	0.001	115.09	0.14	582.30	0.91	12.30	4.00	9.76	1.46	27.36	2.94	4.18	38.59	-
		9	3.92	0.09	0.90	0.008	2.95	0.09	0.05	0.005	121.41	0.58	645.44	3.30	103.44	32.00	79.10	4.87	258.92	7.94	36.69	41.75	-
	AD-P	1	1.06	0.05	0.75	0.043	0.30	0.01	0.01	0.003	110.44	0.13	728.10	1.25	14.10	5.00	4.30	1.66	23.24	3.21	2.90	42.68	16.53
		9	3.83	0.07	0.99	0.009	2.77	0.07	0.05	0.01	116.00	0.56	781.68	2.59	84.82	22.14	70.80	5.97	266.06	9.17	30.57	45.39	5.93

Model / Planner / # Features			Planning Time (s)		ROI Validation (s)		Online CSP (s)		Online MPP (s)		Tour Length (m)		Configurations				Planning Iterations#		Path Evaluations#		Tour Replanned (%)*	Tour time (mins)^	RCE
			$\bar{x}$	SD	$\bar{x}$	SD	$\bar{x}$	SD	$\bar{x}$	SD	$\bar{x}$	SD	$\bar{x}$	SD	New	Removed	$\bar{x}$	SD	$\bar{x}$	SD			
	AD-R	1	18.02	4.17	0.07	0.001	12.10	0.70	5.74	4.13	117.40	0.57	731.94	7.10	729.94	717.00	14.80	10.70	906.88	71.64	100.00	43.96	-
		9	22.86	4.31	0.08	0.001	14.98	0.72	7.69	4.29	119.70	0.70	773.70	6.82	771.70	717.00	18.10	9.89	1,000.74	60.82	100.00	45.74	-
6x6m	AD-P(R)	1	1.35	0.02	1.00	0.009	0.34	0.02	0.005	0.001	162.13	0.14	791.38	0.85	13.38	4.00	9.86	1.29	28.68	2.53	3.21	53.40	-
		9	4.57	0.12	1.17	0.01	3.31	0.12	0.07	0.009	169.23	0.54	866.56	3.23	120.56	36.00	81.32	6.82	275.18	10.37	28.83	57.09	-
	AD-P	1	1.34	0.02	1.05	0.01	0.29	0.01	0.01	0.001	156.50	0.19	1,005.14	0.76	8.58	2.44	8.28	1.68	26.54	2.79	2.11	59.59	30.94
		9	4.33	0.07	1.32	0.01	2.92	0.07	0.07	0.01	162.33	0.48	1,060.16	2.68	84.38	23.22	87.74	6.32	285.14	10.66	24.09	62.39	10.77
	AD-R	1	43.19	6.05	0.10	0.003	20.23	1.21	22.71	5.84	166.54	0.69	1,023.00	8.23	1,021.00	997.00	43.88	11.31	1,434.30	56.23	100.00	61.86	-
		9	47.53	5.33	0.11	0.002	23.50	0.98	23.76	5.36	168.89	0.71	1,063.92	7.36	1,061.92	997.00	45.60	10.36	1,508.52	64.33	100.00	63.61	-
7x7m	AD-P(R)	1	1.73	0.03	1.34	0.02	0.39	0.02	0.01	0.001	217.15	0.14	1,031.72	0.86	13.72	4.00	10.68	1.58	30.00	2.54	2.50	70.58	-
		9	5.35	0.13	1.51	0.01	3.74	0.13	0.07	0.007	224.81	0.59	1,111.76	3.22	125.76	36.00	79.62	4.63	278.68	8.67	22.94	74.53	-
	AD-P	1	1.79	0.02	1.42	0.01	0.36	0.02	0.01	0.001	211.27	0.17	1,363.08	0.97	10.08	3.00	7.18	1.16	26.72	2.12	1.91	80.65	38.96
		9	4.99	0.09	1.74	0.03	3.17	0.09	0.06	0.005	216.66	0.37	1,410.16	2.23	77.96	23.80	74.50	3.49	271.84	4.96	18.60	83.12	16.43
	AD-R	1	70.94	7.91	0.14	0.003	31.79	1.61	38.80	7.74	222.91	0.71	1,345.26	8.03	1,343.26	1,354.00	50.26	9.67	1,918.68	54.09	100.00	81.99	-
		9	82.77	11.21	0.15	0.003	36.44	1.75	45.96	11.33	225.42	0.99	1,390.34	9.04	1,388.34	1,354.00	56.20	12.91	2,014.28	65.11	100.00	83.91	-
8x8m	AD-P(R)	1	1.85	0.03	1.48	0.02	0.36	0.02	0.01	0.001	280.16	0.18	1,304.04	0.81	14.04	4.00	10.14	1.90	29.72	3.10	2.00	90.16	-
		9	5.30	0.12	1.63	0.01	3.57	0.11	0.07	0.007	288.24	0.50	1,383.04	2.87	125.04	36.00	81.80	4.49	284.44	6.59	18.38	94.14	-
	AD-P	1	1.86	0.03	1.52	0.02	0.32	0.02	0.01	0.006	275.68	0.14	1,741.52	0.81	7.52	4.00	8.96	2.76	32.68	5.56	1.29	104.00	58.81
		9	4.93	0.08	1.78	0.02	3.05	0.06	0.09	0.009	281.59	0.44	1,793.46	2.32	79.82	24.36	77.64	4.60	283.42	8.28	14.20	106.71	24.13
	AD-R	1	111.84	12.64	0.15	0.003	40.03	2.08	71.44	12.84	290.55	0.83	1,744.16	11.26	1,742.16	1,736.00	69.38	12.18	2,576.52	67.78	100.00	106.56	-
		9	120.69	17.84	0.16	0.002	44.09	1.74	76.22	17.74	292.48	1.00	1,784.58	9.14	1,782.58	1,736.00	73.08	17.03	2,649.60	75.35	100.00	108.23	-

ROI – Region of Interest *CSP – Coverage Sampling Problem* *MPP – Multi-goal Planning Problem*

 # Combined total over all MPPs for AD-P and AD-P(R) **Number of configurations in final tour replanned. Excludes start and end position* *^Tour time = 2s*Configurations+ 0.1m/s*Tour Length*

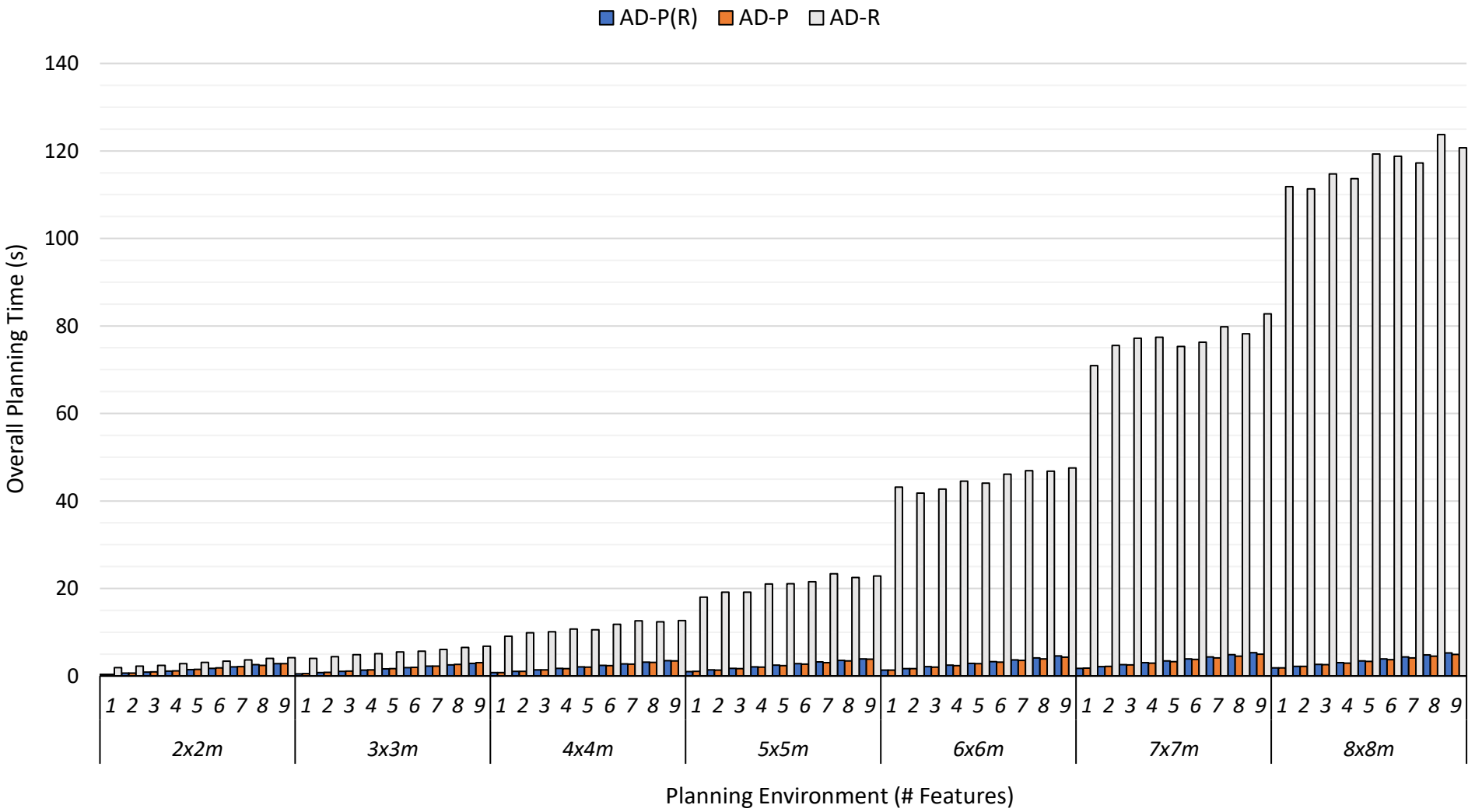


Figure 8-10: Overall planning times for AD-P(R), AD-P and AD-R.

Analysis of the Replanning Times

Figure 8-10 shows the differences between overall replanning times between AD-P(R), AD-P and AD-R. As the planning environments grew larger in size, the scale by which the replanning times for AD-R grew was significantly greater compared to the replanning times of AD-P and AD-P(R). Overall, replanning times for AD-P and AD-P(R) were relatively consistent across all planning problems.

The results demonstrate that the replanning times of AD-P and AD-P(R) increase in proportion to the number of new primitives in the environment. As the same cylindrical feature is added into the environment, a similar trend appears across each planning series. Replanning times are only minimally sensitive to the number of primitives in the environment. The largest difference between replanning AD-P and AD-P(R) times was 8% for planning problem *8x8m_9*, which is a marginal difference considering less than a second separates the replanning times for the largest planning problem. For the more complex planning problem, *2x2m_9*, the replanning times were equivalent.

Relative Performance between Adaptive Coverage Planners

A relative comparison of computational performance between AD-P and AD-R is shown in Figure 8-11. All planning problems suggest a statistically significant result ($p < 0.001$, $d > 0.8$). Across all planning problems, AD-P strongly outperforms AD-R. Computational times were less for large environments for the same number of features as more of the initial tour was preserved. The largest planning problem with minimal environmental influence, *8x8m_1* (1.3%), resulted in replanning times 60.2 times faster than the AD-R. At the other end of the planning problem spectrum, *2x2m_9* (84.6%) was only 1.48 times faster than AD-R. A comparison between AD-P(R) and AD-R, illustrated in Figure 8-12, yielded similar results ($p < 0.001$, $d > 0.8$). For AD-P(R), replanning times for *8x8m_1* (2.0%) were 58 times faster than AD-R and for *2x2m_9* (94.2%) replanning was 1.2 times faster.

Distribution of Overall Planning Times among Planning Processes

Figures 8-13 to 8-15 illustrate the processing time for each *adaptive coverage planner*. Due to the simplicity of path planning within these environments, AD-P and AD-P(R) are more dependent on the ROI Validation and Online CSP processes. As the replanning effort decreases, the ROI Validation becomes the more dominate process because fewer samples are required to cover the new features, hence decreasing the replanning effort. Introducing a *segmentation process* has reduced the use of the CSP and MPP processes that dominate

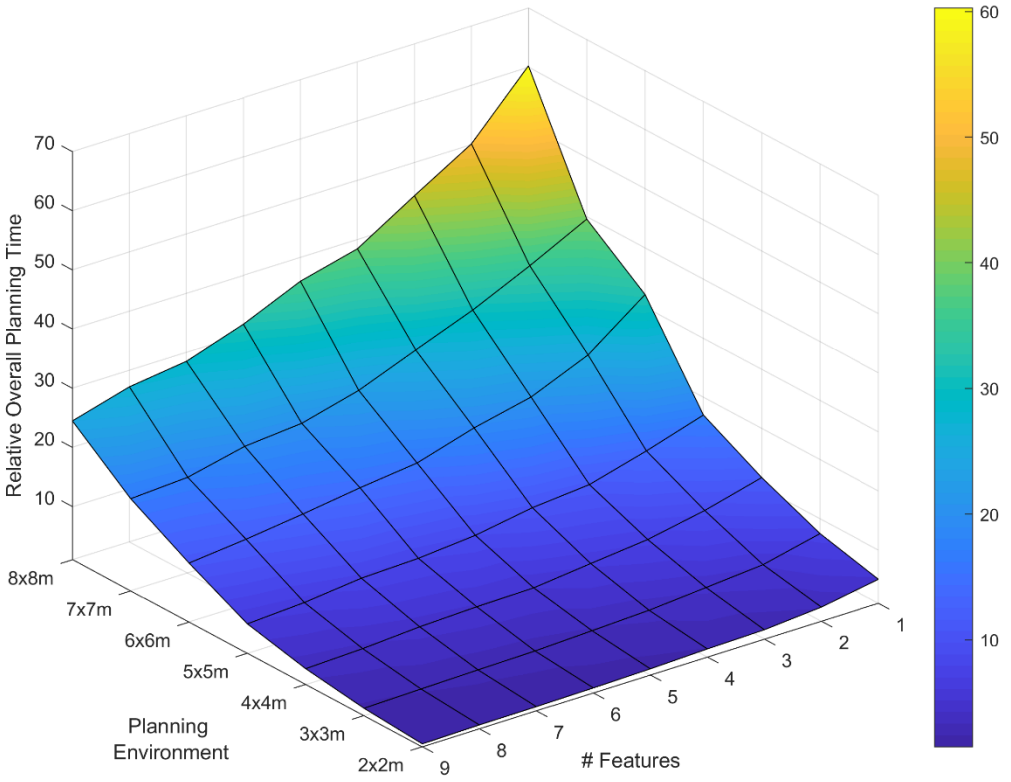


Figure 8-11: Relative comparison between overall planning times of AD-P and AD-R.

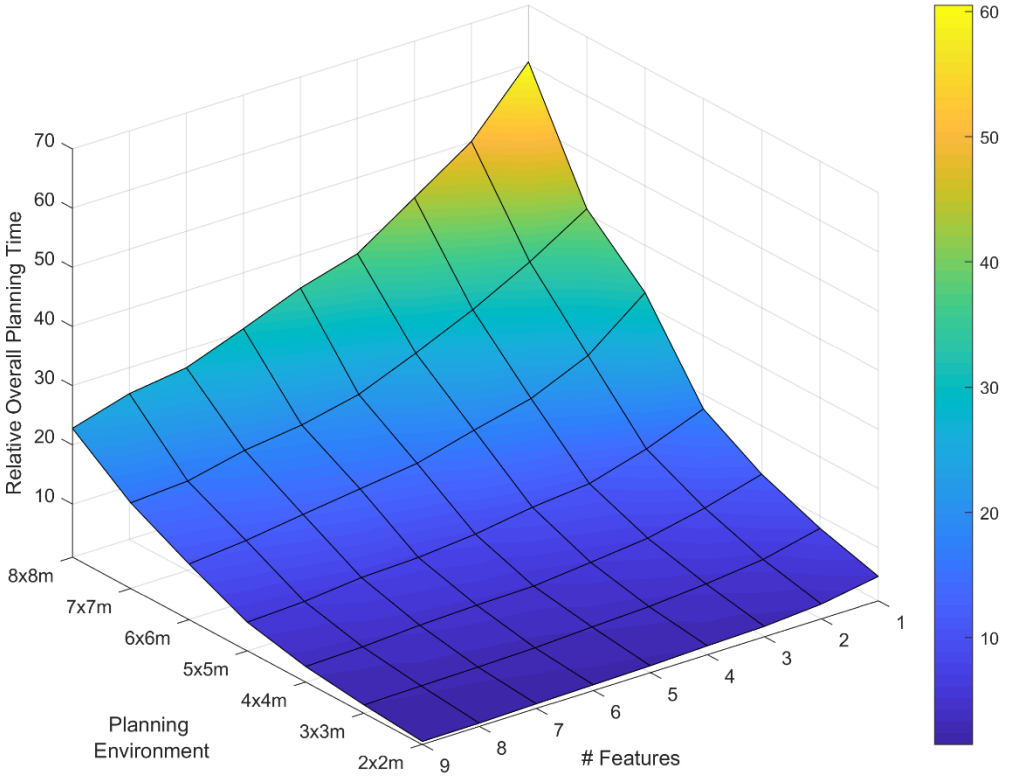


Figure 8-12: Relative comparison between overall planning times of AD-P(R) and AD-R.

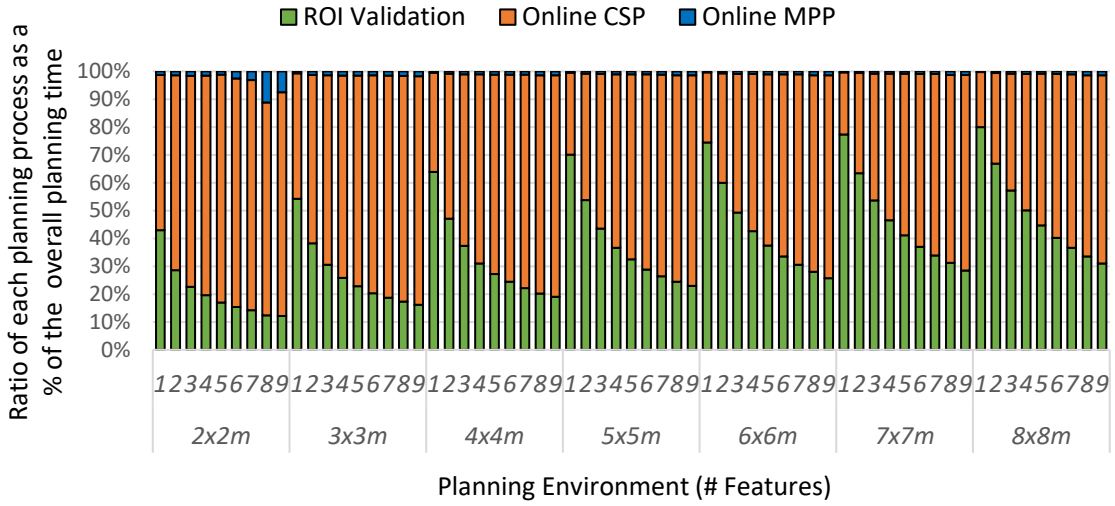


Figure 8-13: Ratio of each planning process for AD-P(R) as a percentage of the overall planning time.

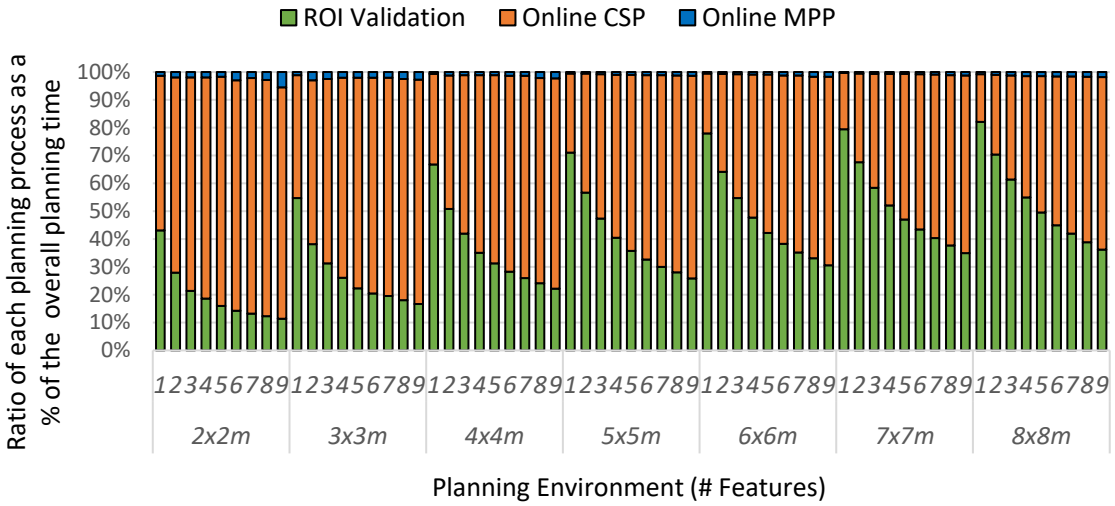


Figure 8-14: Ratio of each planning process for AD-P as a percentage of the overall planning time.

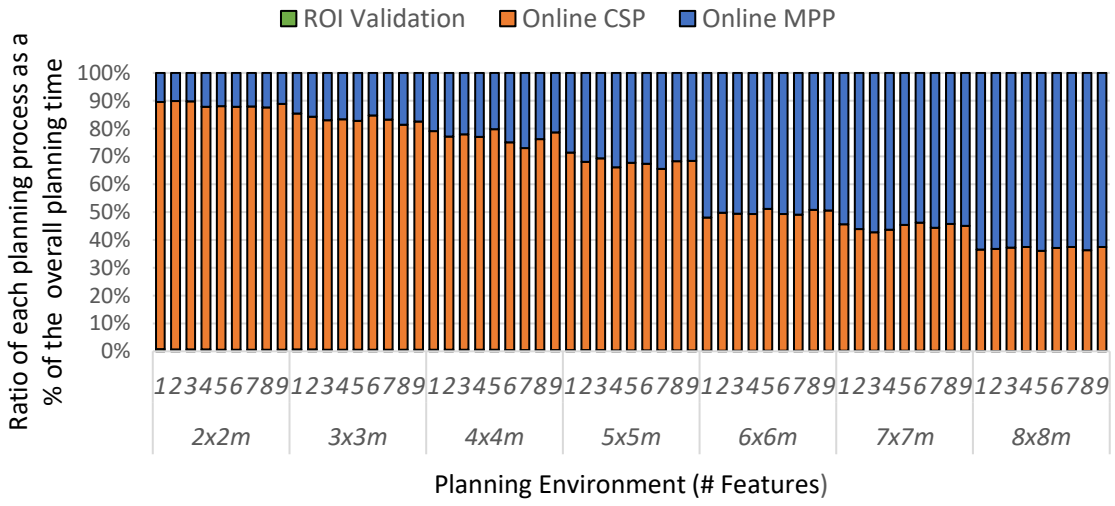


Figure 8-15: Ratio of each planning process for AD-R as a percentage of the overall planning time.

AD-R planning times.

With the statistically consistent covering sets produced by solving the Online CSP, solving the Online MPP has minimal impact on planning times. The only occurrences where Online MPP was more than 5% of the overall planning time, was for planning problems *2x2m (8,9)* when both the AD-P and AD-P(R) replanned more than 75% of the final tour. Despite AD-P and AD-P(R) being initialised with different initial tours, the *plan repair strategy* shows similar trends in the allocation of each of these planning processes to solve the replanning problem.

In comparison to AD-R, the more expensive Online CSP and Online MPP processes dominated planning times. The larger the coverage planning problem became, the more influence the Online MPP process had on the planning times (Figure 8-15). The distribution of planning times demonstrated how the introduction of a pre-processing phase, to localise the replanning effort (ROI Validation), resulted in a reduction in replanning times.

Visual Demonstration of Computational Improvements for AD-P and AD-P(R)

Figure 8-16 presents a visual comparison between replanning methodologies for planning problem *4x4m_5*. The progression of both replanning strategies through the ROI Validation, Online CSP and Online MPP phases clearly illustrate the efforts each adaptive coverage planner requires to perform a replanning update (Figure 8-16b-d). The visual comparison also demonstrates the ability of the AD-P to replan multiple ROIs concurrently within the same replanning iteration (Figure 8-16d). Fewer configurations and paths were replanned for AD-P and AD-P(R) compared to AD-R, where the *full replan strategy* resulted in several hundreds of new configurations and paths (Figure 8-16d).

Reduction of Configurations, Path Evaluations and Planning Iterations

The reduction of the overall computational times was due to;

1) The significant reduction of new configurations to cover the new features ($p < 0.001$, $d > 0.8$).
2) The significant reduction in the number of path evaluations needed to solve each MPP ($p < 0.001$, $d > 0.8$). Only *2x2m_9* for AD-P was the only planning problem that evaluated more paths than AD-R.

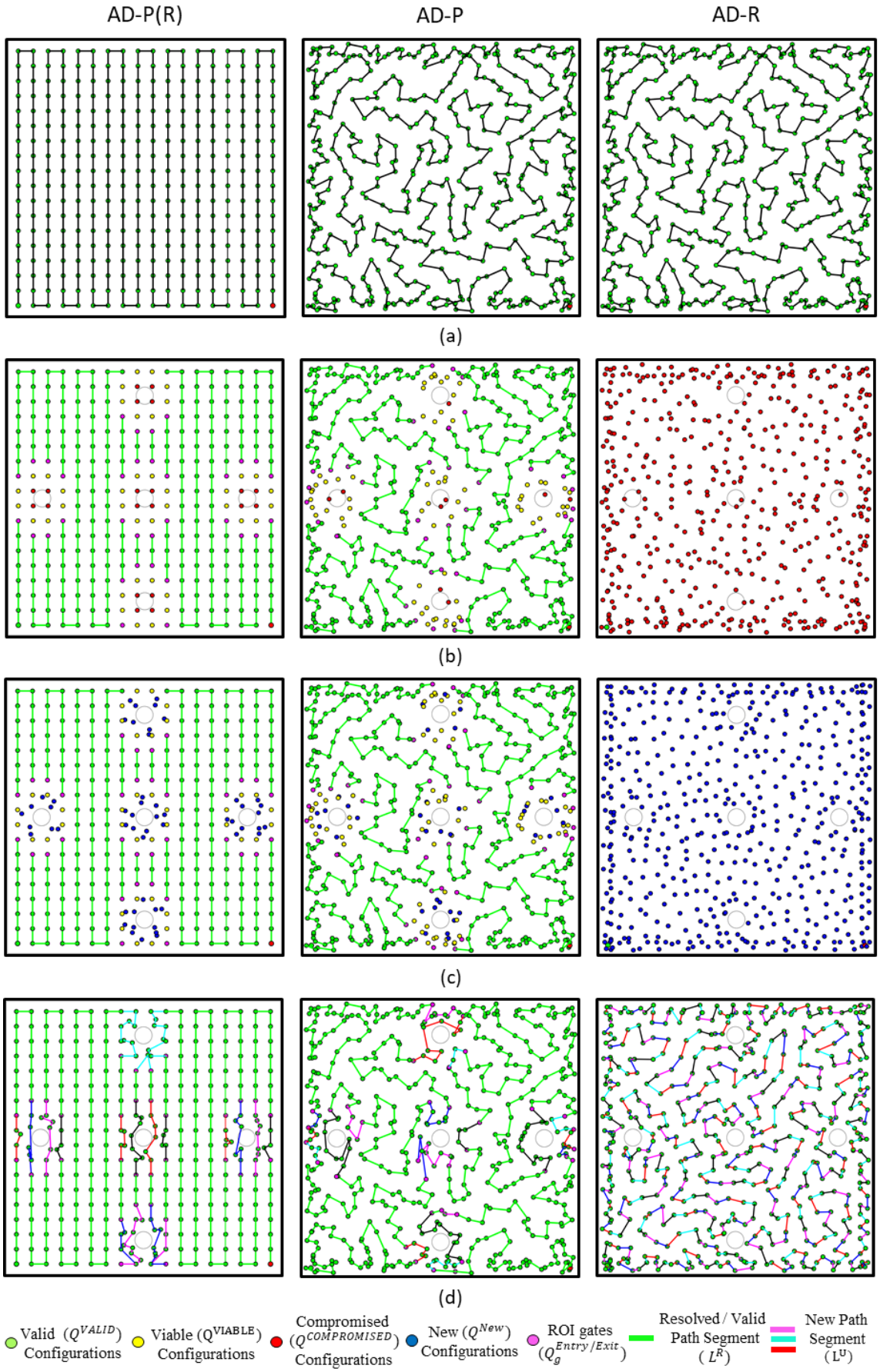


Figure 8-16: A visual comparison between AD-P, AD-P(R) and AD-R. (a) Initial offline tour. (b) ROI Validation phase. (c) Online CSP phase. (d) Online MPP phase and resultant final tour.

Figures 8-17 and 8-18 compare the number of new configurations introduced into the new tours and the number of paths AD-R and AD-P evaluated for each planning problem. Both the *full replan strategy* and the *plan repair strategy* proportionally increased with the number of new primitives each replanning strategy was required to replan. Due to only replanning within the ROIs, AD-P generated smaller set sizes and therefore consequently evaluated fewer paths. As a result, the trends witnessed in both Figures 8-17 and 8-18 reflect the same trends observed for both *adaptive coverage planners* with respect to overall planning times (Figure 8-10).

As the ROIs for each new primitive were identical for both AD-P and AD-(R), there was a similar number of new configurations introduced into each covering set. The differences between how many new configurations were required depended upon;

1) the number of existing configurations located within the ROIs that require recalculation, and
2) the random nature of the sampling process.

As the results indicate, AD-P(R) replanned more of the existing tour than AD-P (Table 8-3). As more of the existing configurations became candidates for replanning the likelihood of these being replaced by new configurations increased. Figure 8-19 illustrates the number of new configurations added for each planning problem against the number of configurations removed from the original tour for both AD-P and AD-R. Not surprisingly, the more configurations removed, resulted in more configurations being required to rectify the loss of coverage. However, while AD-P(R) generally had to produce more new configurations to compensate for the influence of new features, the number of new configurations generated per feature was relatively consistent across all planning problems (Figure 8-20). This behaviour was also observed for AD-P. Therefore, despite the influences the features had on the initial tour, Online CSP continued to scale to the number of new primitives added to the problem.

As discussed previously, the reduction of covering set sizes led to a significant reduction in the number of paths evaluated to solve a given MPP. However, the reduction in path evaluations has not led to the reduction of planning iterations (Figure 8-21). Despite AD-P and AD-P(R) solving the majority of the planning problems significantly faster than AD-R, both AD-P and AD-P(R) collectively performed more planning iterations than AD-P. For AD-R, the number of planning iterations increased with the size of the covering set. It was

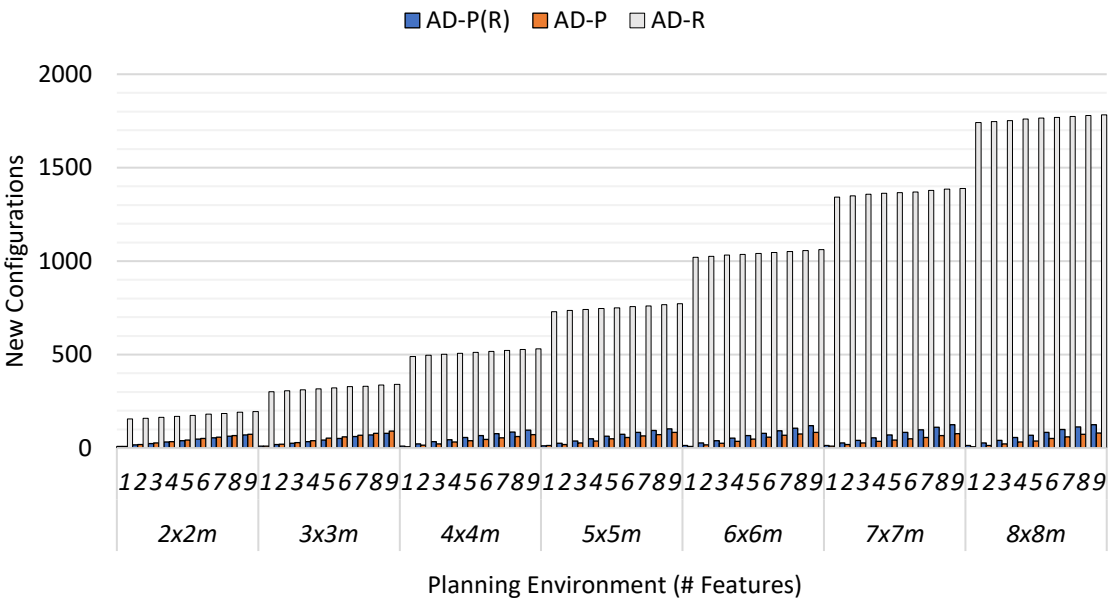


Figure 8-17: New configurations introduced into each planning problem for AD-P(R), AD-P and AD-R.

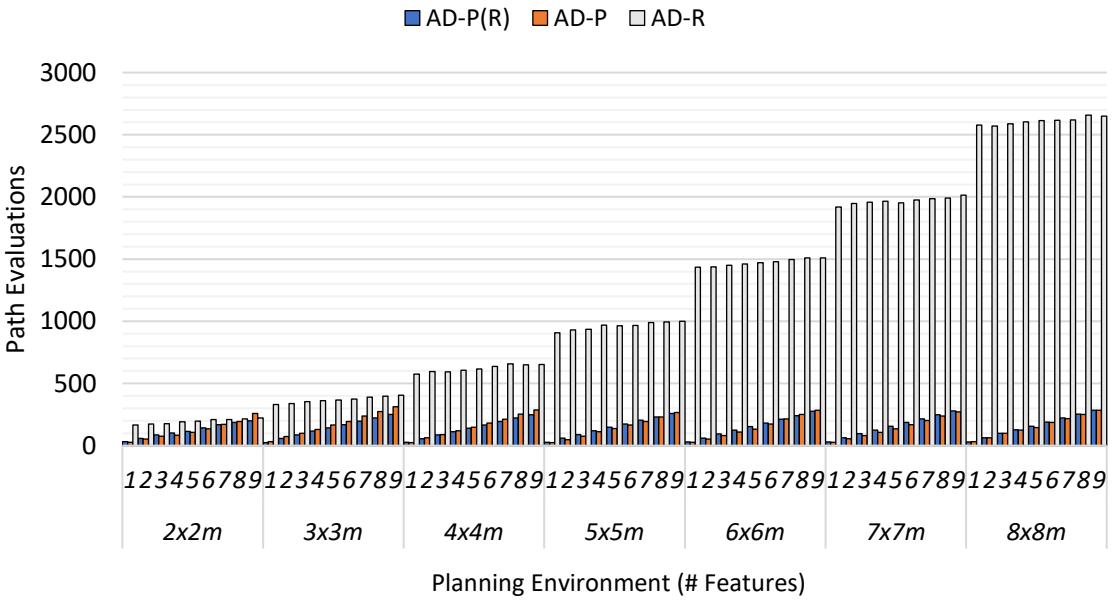
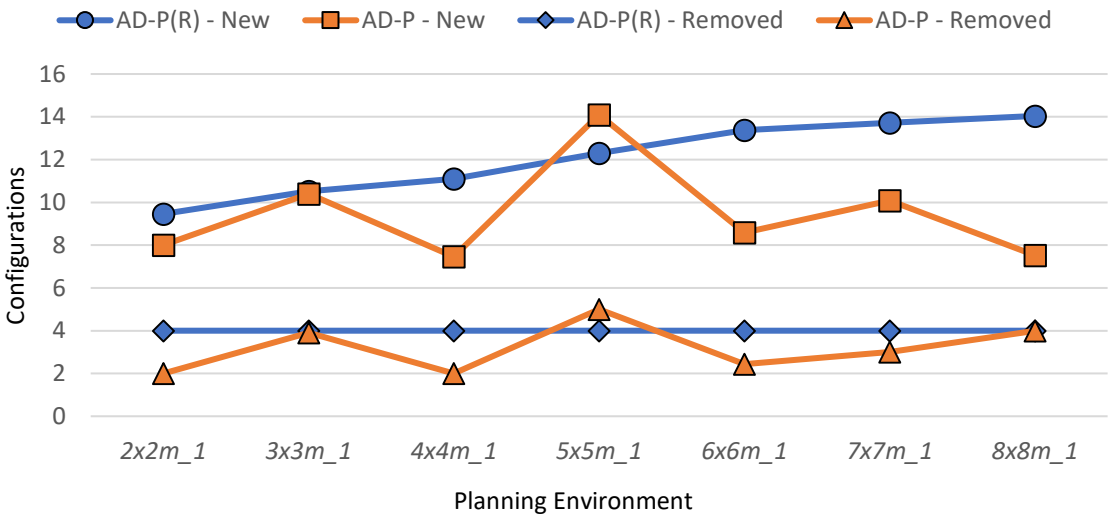
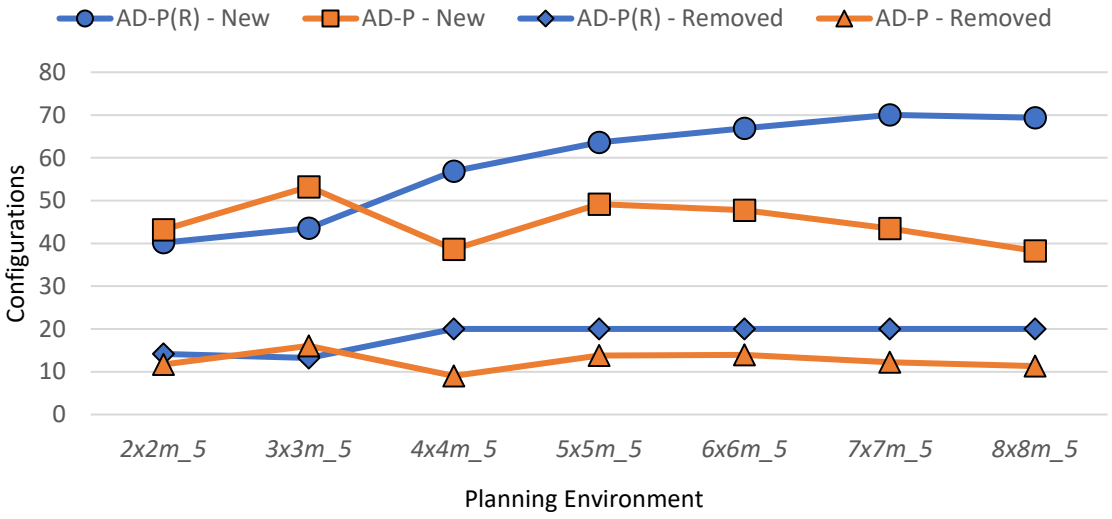


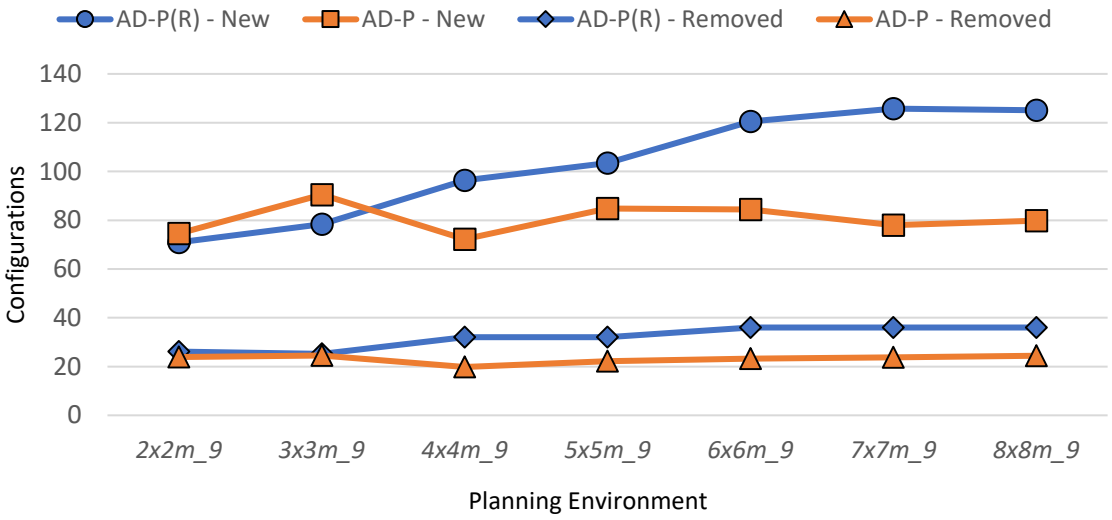
Figure 8-18: Number of path evaluations required to solve the multi-goal planning problem for AD-P(R), AD-P and AD-R.



(a)



(b)



(c)

Figure 8-19: Difference between the number of configurations removed and introduced to solve a planning problem for AD-P and AD-P(R) for (a) one feature, (b) five features and (c) nine features.

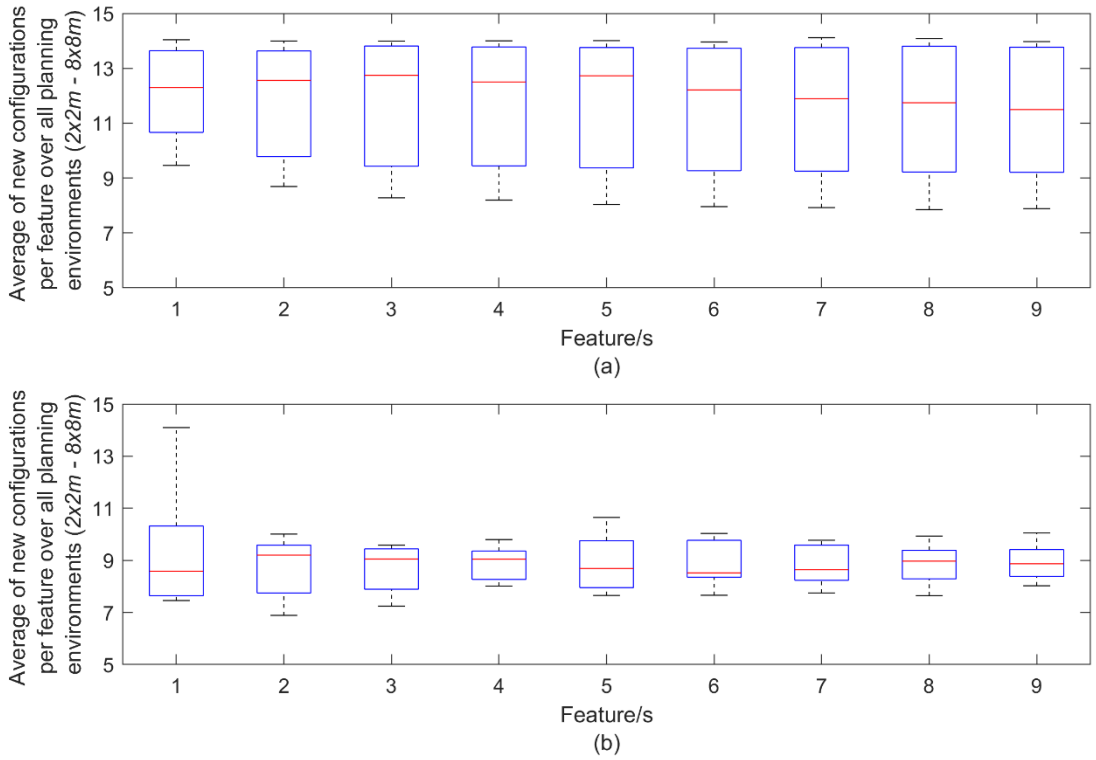


Figure 8-20: Despite the influence of the features on the initial tour, a consistent number of configurations were being generated for each feature over each planning environment for all features. (a) Results for AD-P(R). (b) Results for AD-P.

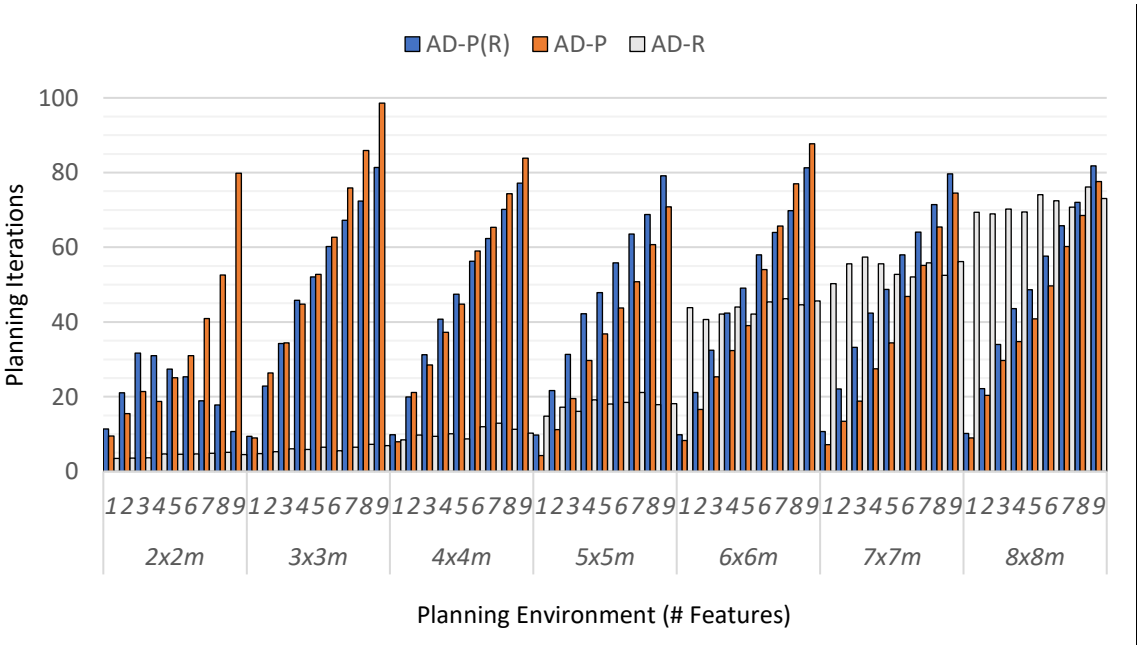


Figure 8-21: Number of planning iterations for AD-P(R), AD-P and AD-R.

not until the covering set sizes were of significant size (1000+) where AD-R performed as many planning iterations as AD-P and AD-P(R).

The trends shown in Figure 8-21 suggest that the number of planning iterations were proportional to the amount of replanning required. There is a difference between the number of planning iterations of AD-P and AD-P(R) for planning environments *2x2m(6-9)* but this is due to a separate issue that is commented on later in the discussion.

Despite the significant increase in planning iterations for the *plan repair coverage planners*, the additional computation has a negligible impact on planning times. While it was not confirmed in this analysis why this behaviour occurred, a possible suggestion could be due to the *state of minimal improvement* and the minor variability that was present before the LPP is terminated (Section 5.6.1; Table 5-10).

As there are several sub-MPPs being solved, each one could potentially iterate for a few extra iterations after entering the *state of minimal improvement* before terminating. The accumulation of these additional planning iterations, which are likely *Zero Path Evaluation Iterations* (ZEIs; Table 5-10), could quickly increase the replanning that is required to perform. While the data was not collected in this experiment to prove this assumption, the additional time these additional planning iterations may have had, made no significant impact on Online MPP times.

The potential reason these additional planning iterations would not impact planning times more significantly would be the size of the sub-MPPs; a smaller MPP would not exhaust the time allocated for the TSP solver to find a solution. These smaller TSP problems would solve much faster than those TSP problems formulated by any AD-R covering set. Comparatively, for the larger planning problems, *6x6m* onwards, where the covering sets for AD-P exceeded into the thousands, the TSP solver would have been using the *one second time allocation* to solve the TSP problem while AD-P, with smaller MPPs, would have been using no more than the *0.5 second time limit* allocated (Sections 4.2.4).

As the results show, as the number of planning iterations for AD-P made no significant impact on planning times, they were presumed to be of no issue considering the difference between Online MPP times and the significant reduction in path evaluations. In this context, path evaluations are deemed to be more expensive than planning iterations, especially if the planning constraints become more complex than implemented in these experiments. The

ability of the *plan repair strategy* to reduce the number of times the motion planning is utilised, aligns with the focus of the thesis and the motivations for improvements documented in previous chapters. Given that the *plan repair strategy* achieves this reduction is a successful outcome.

Tour Degradation

A comparison between AD-P and AD-R tour lengths found that tour degradation occurred when more than 35% of the final tour was replanned (Figure 8-22). In total, 16 of 63 AD-P planning problems experienced tour degradation. Only two planning problems, *2x2m_2* and *3x3m_4*, had tour lengths greater than the equivalent AD-R planning problems at a lower replanning effort, 29.7% and 31.6% respectively. Of the 16 solutions that experienced tour degradation, all are from the smaller planning problem series, *2x2m_(2-9)*, *3x3m_(4-9)* and *4x4m_(8-9)*. The *2x2m_1* scenario was the only planning problem from the *2x2m* series not to be included in the set of 16 planning problems that experienced tour degradation, despite 18.2% of the final tour being replanned.

The planning problems that experienced the greatest degradation were *2x2m_9* at 27.6% and *2x2m_8* at 18.7%. Both planning problems experienced the highest replanning effort of all planning problems. The remaining 14 planning problems experienced tour degradation between 0.5% to 10.5%, with replanning efforts between 29.7% to 72.2%.

In contrast, the majority of the tour lengths for AD-P(R) are less favourable compared to AD-P (Figure 8-23). Raster plans, as with the initial tours, are still 1.6% to 9.2% longer than AD-P, despite having smaller covering sets for all planning problems. The tour lengths for *2x2m_(6-9)* were the only occurrences where AD-P(R) produced a better overall tour length than AD-P. *2x2m_9* tour length for AD-P(R) was 27.4% shorter than AD-P with a higher replan effort of 94.2% and 84.6% respectively.

Similar to the AD-P, there were a series of smaller planning problems where AD-R produced tour lengths shorter than AD-P(R). These models included a similar subset but also included *5x5m_(7-9)*, the first of the middle to large planning environments experiencing tour degradation. On average, AD-R generated coverage plans that where shorter by 2.2% to 5.0%. There were four occurrences that AD-P(R), outside the primary range produced shorter plans (Figure 8-23). However, the differences between these planning problems were marginal, no greater than 1%. Unlike AD-P, AD-P(R) for the two most replanned

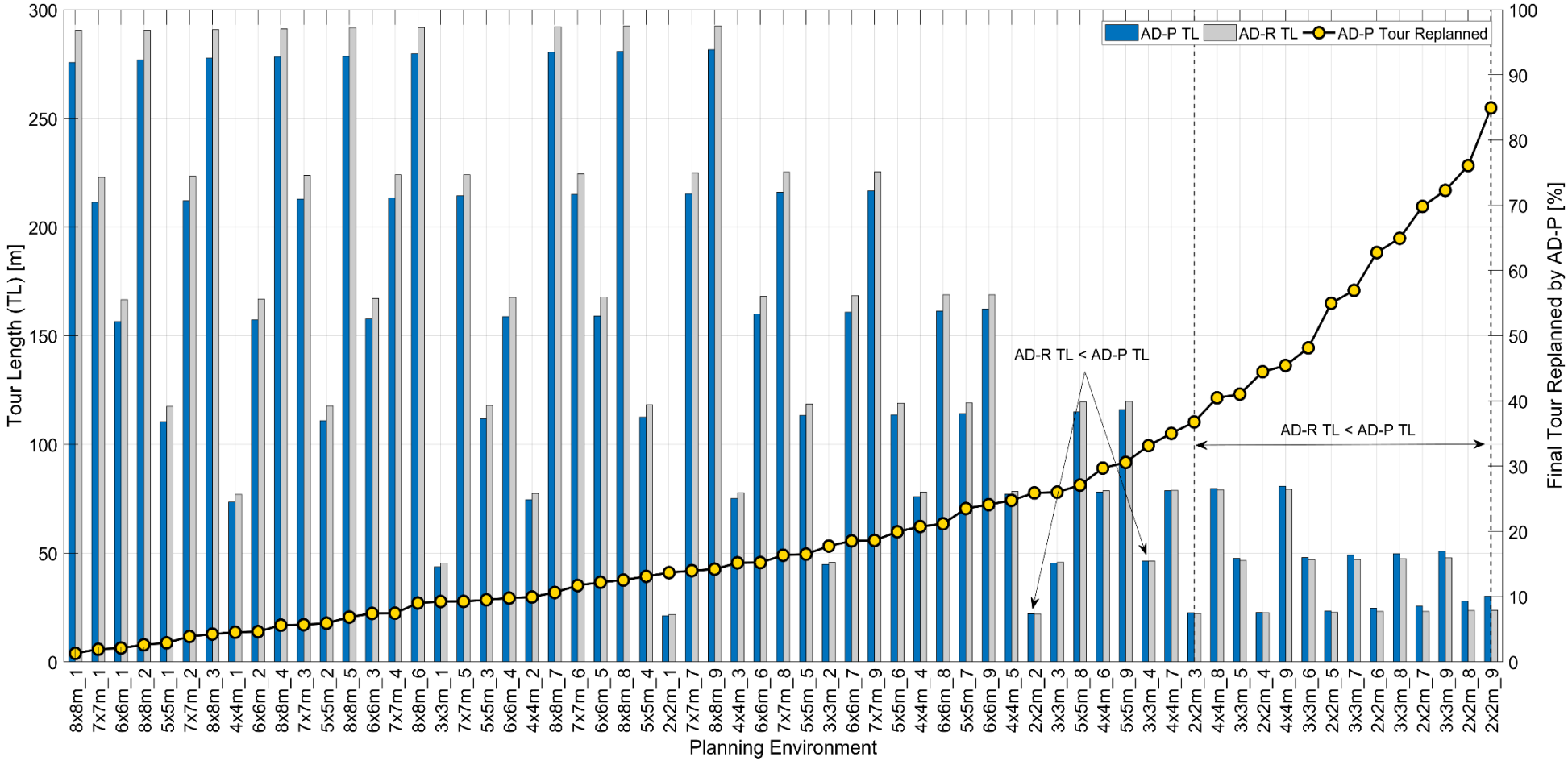


Figure 8-22: The tour degradation of AD-P for the amount of replanning required to perform a tour update.

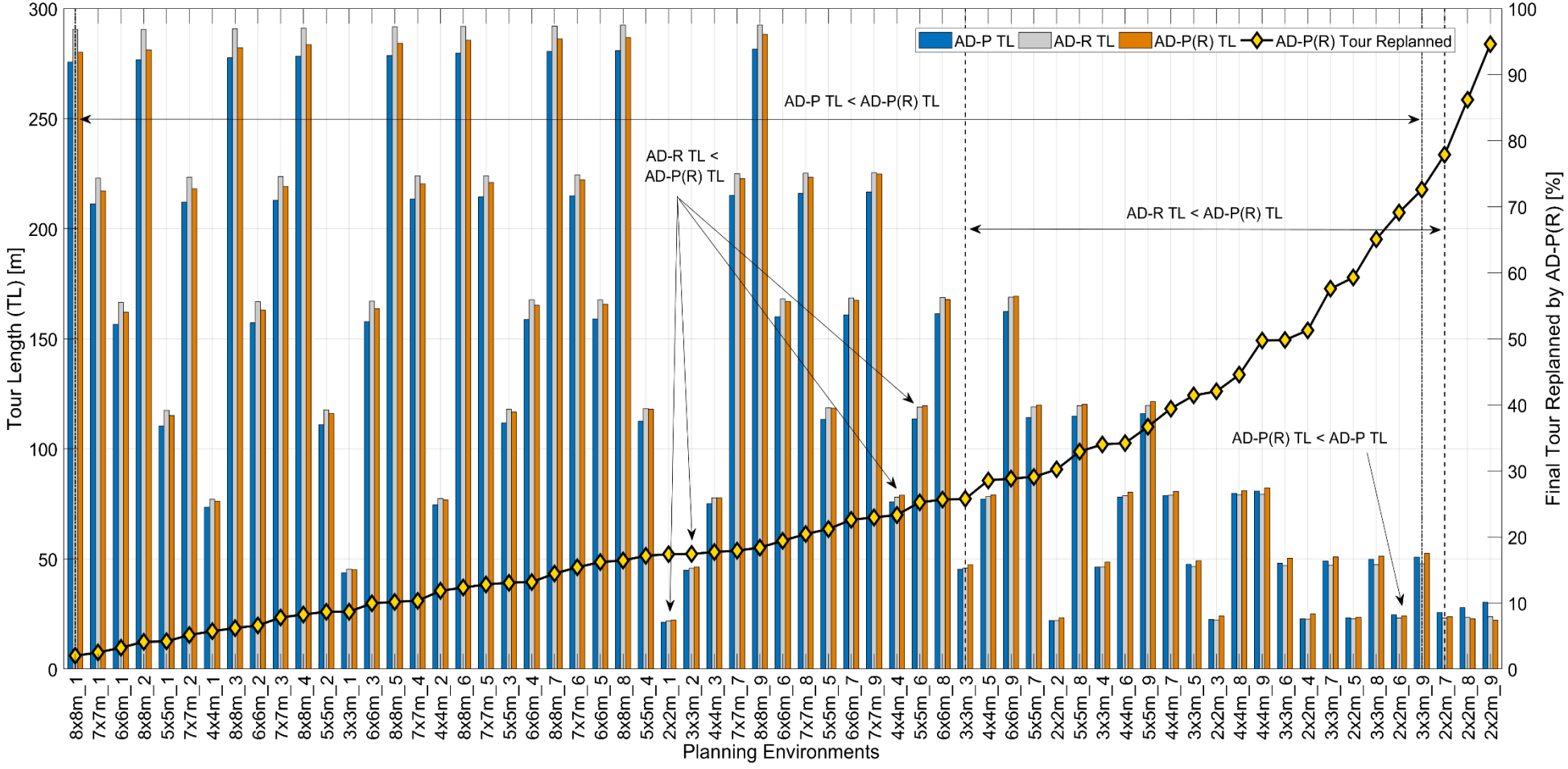


Figure 8-23: The tour degradation of AD-P(R) for the amount of replanning required to perform a tour update.

environments *2x2m_(8-9)* produced tour lengths of better quality.

Tour Time and Relative Computational Effort

Despite experiencing a greater tour degradation in comparison to the other planners, AD-P(R) *tour times* (Equation 8-1) were fastest overall. As the tour degradation for AD-P was not as significant as expected, the relative *tour time* between AD-P and AD-R were relatively the same across the majority of the planning problems (Figure 8-24). The smaller coverage sets of AD-R produced in the smaller planning problems were not enough to produce a better *tour time* for these problems. *Tour times* for AD-P planning problems were only up to 3.5% less than AD-R. The only AD-R planning problems that were better than AD-P coincided with the smaller planning problems, *2x2m_(1-9)*, *3x3m_(2-9)*, *4x4m_(5-9)*, whose replan effort was higher than 35% and tours were either longer or similar in length to AD-R (Figure 8-22).

With similar *tour times*, the RCE between AD-P and AD-R, as illustrated in Figure 8-25, yielded a similar trend to what is shown in Figure 8-12. This result suggests that, computationally, AD-P is a better overall strategy to employ. In all cases below a replan effort of 85%, the trade-off of tour quality is not significantly degraded compared to the computational speed-up that can be achieved by employing AD-P to solve the online planning problem.

8.3.2 Discussion

The aim of this experiment was to analyse the response of the AD-P by controlling the level of change in the problem. This enabled the range of operational limits to be determined within which the *plan repair strategy* would be most effective. In summary, the results of this experiment have shown that segmenting the initial tour before replanning has resulted in all tested planning problems being solved faster than the *full replan strategy*.

Does the adaptive partial coverage planner scale to the size of the environment or the number of additional primitives?

The results of the experiment confirm that the *plan repair strategy* using ROIs does successfully constrain the replanning effort to parts of the tour affected by changes in the environment. The AD-P and AD-P(R) showed a minimal difference in replanning times despite having two different initial offline tours. The *plan repair strategy* created smaller covering sets, therefore creating smaller MPPs. The focus on reducing the size of the problem has led to a significant reduction in the number of new configurations and path

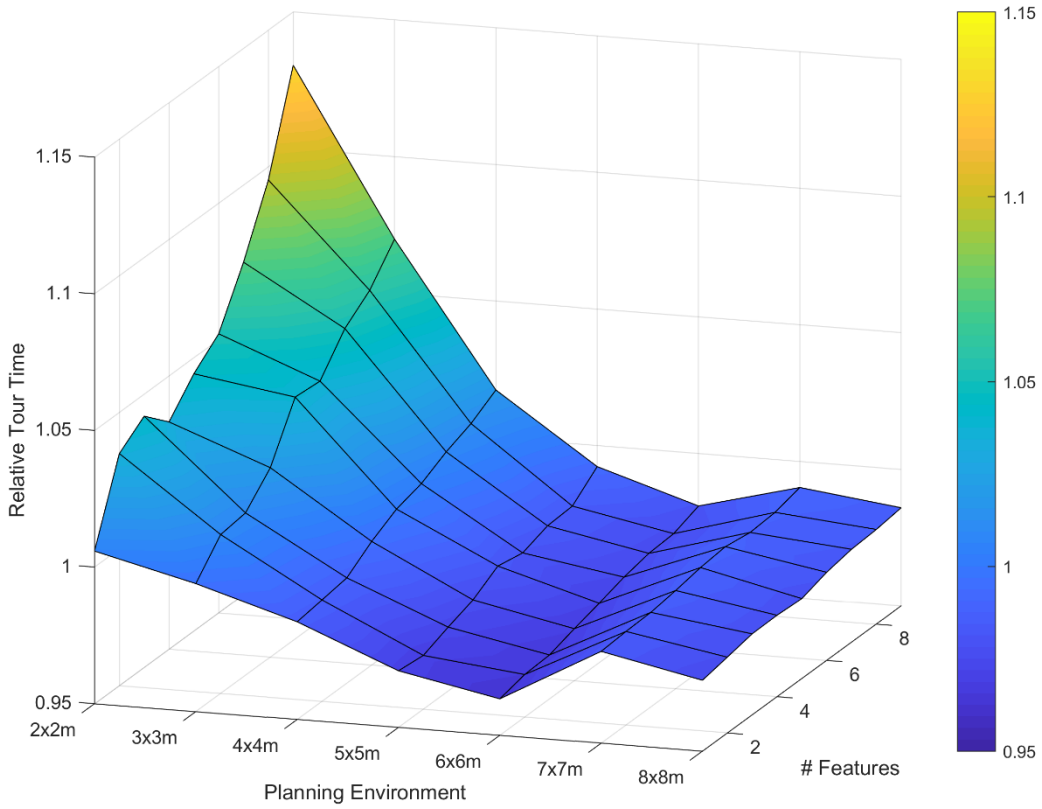


Figure 8-24: Relative *tour time* between AD-P and AD-R.

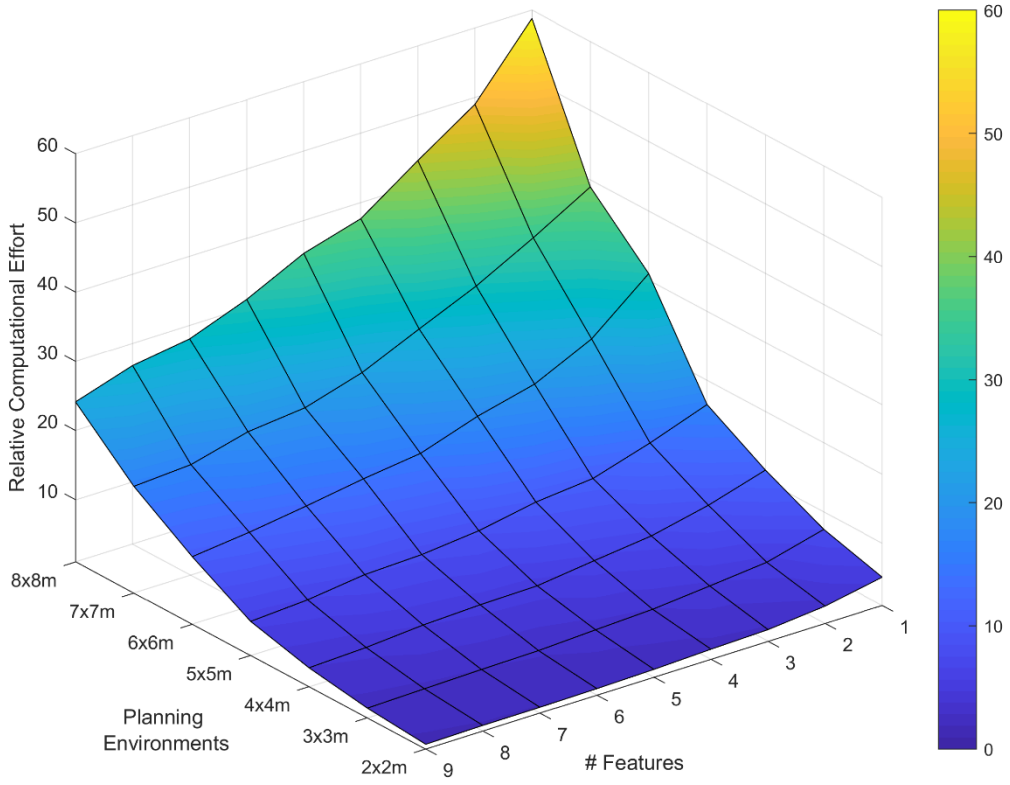


Figure 8-25: *Relative computational effort* between AD-P and AD-R.

evaluations required to solve the online replanning problem. A higher replanning effort was required by AD-P(R) to provide an updated tour as more configurations were removed and subsequently replaced in the tour. As the same feature was introduced, it was expected a similar number of configurations would be required to cover each feature regardless of the initial tour (Section 8.2.3). As expected, both AD-P(R) and AD-P provided a consistent number of new configurations to cover the new features (Figure 8-20). This also results in consistent and repeatable tour lengths which are critical for online applications.

What was the computational impact of segmentation and merging of sub-plans on the planning times?

In these experiments, the computational impact of the segmentation and merging processes were negligible compared to the replanning times of AD-R (Table 8-3; Figure 8-10). The cost of solving larger MPPs far outweighs the cost of the *segment* and *merge* processes. The ROI Validation scales to the number of configurations of the current tour while the *full replan strategy* scales to the size of the environment. ROI Validation times between AD-P and AD-P(R) demonstrates this difference, whilst computationally negligible (Table 8-3).

Despite the ROI Validation phase dominating the majority of the planning problems for AD-P and AD-P(R), the overall planning time was significantly reduced by the overwhelming reduction of Online CPP and Online MPP times due to the segmentation process of ROI Validation. As planning problems are smaller and path planning is relatively simple within these environments, solving the CSP became the more dominant of the two main planning processes (Figures 8-13 and 8-14).

The results showed that computationally, the ROI Validation phase times will increase with the size of the initial tour and the complexity of the robotic mobility and visibility constraints. Additionally, the more complex the environment becomes, the more computationally expensive the Online CSP and Online MPP will become.

How much replanning of the tour would need to occur to allow AD-P to be just as expensive as replanning using AD-R?

This experiment did not, for the environments tested, determine the exact amount of replanning that would significantly degrade both the replanning times and tour quality of AD-P enough that would render AD-R to be a better replanning strategy. All replanning times for AD-P and AD-P(R) were less than AD-R, even at the highest recorded replanning efforts of 84.9% and 94.6% for each AD-P and AD-P(R), respectively. At these limits, with the exception of *2x2m_(7-9)*, these *adaptive coverage planners* exhibited minimal tour

degradation that would warrant the use of AD-R in this scenario. In these cases of *2x2m_(7-9)*, despite the tour degradation present, there were significant computational savings by AD-P as demonstrated in the RCE, with the worst case being 2x2m_9 having a RCE value of 1.68.

What is the potential drop in tour optimality as the sub-plan approach does not solve the coverage plan globally?

It was expected this experiment would demonstrate that as a compromise for shorter replanning times tour degradation would occur due to the *plan repair strategy*. *It was surprising that the experiment demonstrated that tour lengths for AD-P and AD-P(R) were shorter than AD-R and consequently tour degradation predominately occurred after replanning 35% of the final tour*. It was expected that tour degradation would occur earlier since the *plan repair strategy* only has the ability to replan locally while the *full replan strategy* replans globally. However, the analysis of the data showed, at least for the tested scenarios, that the final tours *do not support this hypothesis*.

From the analysis, it seems the degradation of the AD-P and AD-P(R) tours did not occur due to:

1) The nature and impact of the new objects had on the initial tours.
2) Changing the redundancy of the *redundant roadmap* when solving the Online CSP.

Tour degradation for the majority of the AD-R planning problems was due to replanning *redundant roadmaps* with a lower level of redundancy than what was used for the initial offline tour. A *redundancy-five roadmap*, as opposed to *redundancy-ten*, used for the offline tours, was chosen using the OSP to reduce the computational impact of the Online CSP process (Section 8.2.7).

As proven by Englot and Hover (2017), *redundant roadmaps* of higher redundancy create shorter tours. As the *full replan strategy* removes the remaining tour, the offline tour created by the *redundancy-ten roadmap* is replaced with a replanned tour created by a *redundancy-five roadmap*. Given the OSP produced fewer configurations for a *redundancy-five roadmap* than that generated by the *random sampling procedure*, it could have been expected that AD-R could produce tours of equivalent quality to AD-P (Section 8.2.7). However, for the planning problems that required minimal replanning, the AD-R did not produce better quality tours as was expected. The results showed that while the OSP is able to generate smaller covering sets than that of the *random sampling procedure*, the covering

sets for AD-R were not better in quality than the tours preserved by the *plan repair strategy*.

The lack of AD-P expected degradation was due to the ability of the *plan repair strategy* to preserve the majority of the existing tour that was constructed from a *redundancy-ten roadmap*. As the environments increased in size with fewer features, the relative replanning effort naturally decreased. Consequently, more of the initial tours were being preserved. Given that majority of the plan was preserved, the minor replanning that did occur did not significantly impact the overall tour quality. Therefore, it can be concluded that if AD-R were to use a higher redundancy to construct *redundant roadmap*s, it would potentially result in tour lengths of equal and better quality than AD-P but would come at the expense of longer Online CSP times.

Scenarios when tour degradation occurred in AD-P relative to AD-R

While a majority of the planning problems did not suffer tour degradation, interesting findings were observed from the minority of the AD-P and AD-P(R) planning problems that did suffer tour degradation. When the replanning effort exceeded 35%, tour quality for AD-P and AD-P(R) began to degrade as initially expected. Examining the tours of AD-P for these planning problems uncovered two limitations of the *plan repair strategy*;

Limitation 1) The new tour must follow the strict ordering of the existing tour despite changes in geometry.

Limitation 2) The sorting algorithm used to allocate ROI configurations into sub-MPP does not factor a geometrical representation of the environment to determine which path segment is geometrically appropriate.

Figure 8-26 provides a visual example of these two limitations AD-P present for its most challenging planning problem *2x2m_9*. Given the changes in the environment, the order of the initial random tour is no longer suitable for the new geometry. The influence of all the changes created several ROI gates (Section 7.6) along the perimeter of the environment (Figure 8-26a). As the same tour direction must be preserved, new tour segments must be routed through these preserved segments. Given the impact of the changes creates several gate parings that may not be appropriate to produce a suitable solution. This trial created 33 path segments where 14 of these segments required replanning, which included the routing through several single configuration segments, increasing the likelihood of poor configuration placement.

Configurations within a ROI are sorted into sub-MPPs based on the *point-to-line evaluation* between the position of the configuration and all ROI gate pairings (Section 7.6). The *point-to-line evaluation* does not factor how the new geometry impacts the current tour. Placement of the configurations is based on the Euclidean distance to the line, instead of the actual, or an approximated distance. For small localised changes, the sorting algorithm is a sufficient evaluation when small perturbations appear within the geometry of the environment. The results show this to be true for the majority of the planning problems that do not require significant replanning. However, as more of the tour becomes influenced by changes, as seen with *2x2m* planning set, the combination of the sorting algorithm and the preservation of the existing tour order does have the capability to degrade tour quality. By not factoring a geometrical representation into the sorting process, a problem is created that is equivalent to the *Euclidean assumption* underestimating the connectivity between configurations before solving the MPP.

Tour degradation between AD-P and AD-P(R)

Despite the *plan repair strategy* demonstrating the limitations discussed in the previous section, there are examples in the results that indicate that the two limitations improved tour quality, where degradation was expected. There were four planning problems, *2x2m_(6-9)*, where AD-P(R) produced better tours than AD-P and two examples where AD-P(R) produced better than AD-R. All these occurred at the higher replanning efforts between, 69.1% and 94.6%. Considering that all raster plans are longer than the random plans (Table 8-2), it was surprising that the AD-P(R) tours with the highest replan rate succeeded over AD-P and AD-R.

Figure 8-27 shows the impact that nine changes had on AD-P(R), i.e. *2x2m_9*. While AD-P produced a 27.6% increase in tour length compared to AD-R, AD-P(R) reported a 6.6% decrease. Due to the layout of the raster tour, the size of the ROI and the placement of the new features, the preserved path segment and gates for each ROI happened to be conveniently placed for the new tour.

Unlike AD-P, which replanned 16 path segments, this trial replans three path segments, increasing the chances that configurations will be placed in an appropriate segment given the geometry of the environment. As expected, as the replanning effort approaches 100%, the more likely that AD-P and AD-P(R) will produce results similar to AD-R. Given that the tour by AD-P(R) is shorter, AD-P results for *2x2m_(7-9)* were within 1% of AD-R and well

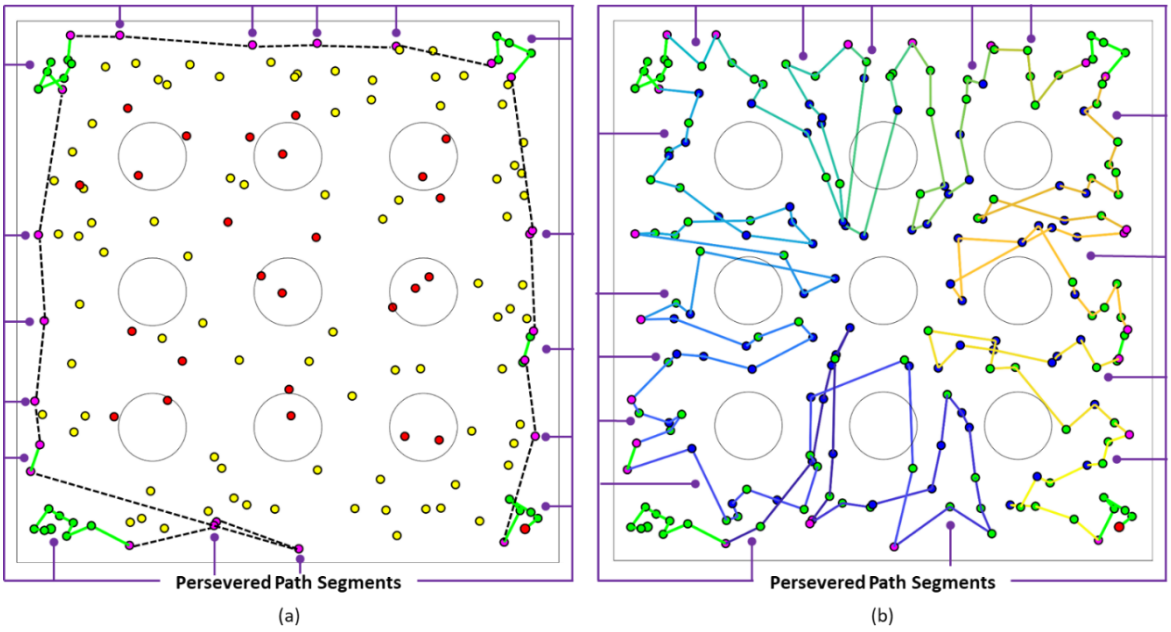


Figure 8-26: Routing through pre-existing tour locations can degrade tour quality. The dashed lines are the point-to-line comparison used between each paired gates.

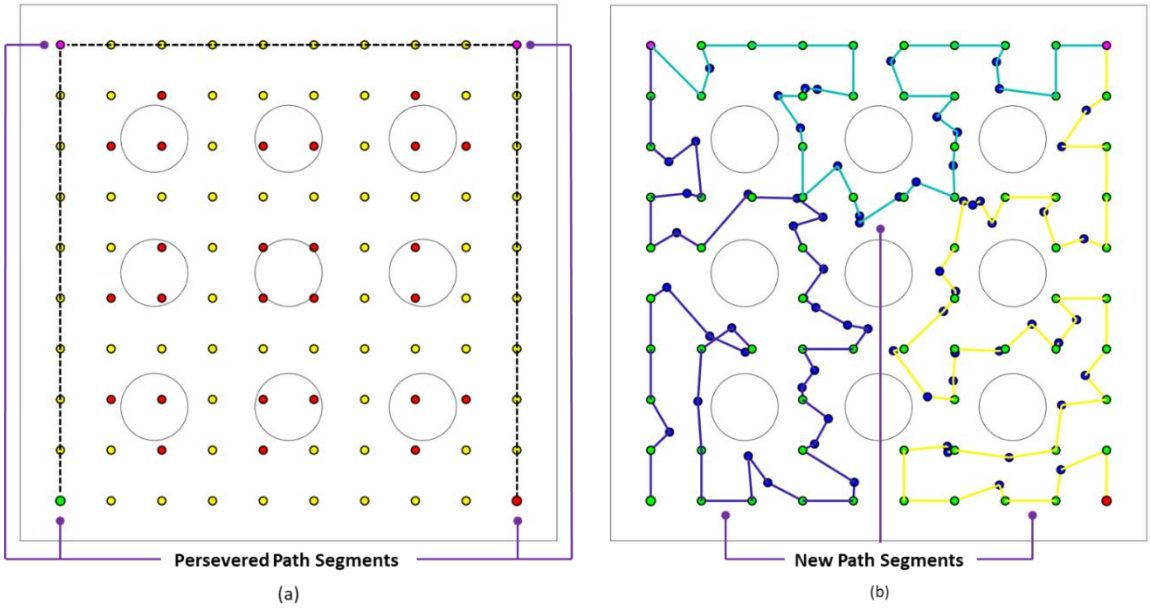


Figure 8-27: Routing through pre-existing tour locations can degrade tour quality. The dashed lines are the point-to-line comparison used between each pair of gates.

well within the variability of the LPP (Section 5.4).

This difference between AD-P and AD-P(R) for *2x2m_(6-9)* highlighted that the *point-to-line evaluation* to sort configurations into *unresolved path segments* may not be suitable for all replanning situations, particularly when the replanning effort is significantly high. The complexity of the original environment, the impact of the new features, and the original tour all play a role in determining the outcome of the sorting process and consequently the replanned tour. Despite the inapplicability for a minority of planning problems, the sorting process does ensure that a tour can exist. Considering the listed limitations, the AD-P was still capable of producing computationally efficient solutions.

8.3.3 Summary

In summary, AD-P was designed to replan only the changes present in a large environment. The results of this experiment have shown that;

1) Implementing a pre-processing phase to segment an existing tour to partially replan the influenced regions has significantly reduced overall replanning times compared to a *full replan strategy*. The AD-P, regardless of the initial tour, was always computationally faster than AD-R. Even at the replanning effort of 94.6%, (*2x2m_9* for AD-P(R)), AD-P still calculated faster.

2) The results have shown the AD-P has met initial expectations. The experiment has clearly demonstrated that the AD-P scales with the size of the changes and not the size of the environment, unlike AD-R. The number of new configurations and path evaluations required to solve the online replanning problem is consistent for solutions provided by AD-P and AD-P(R). Any variation can be attributed to the influence the changes have on the initial tour and the segmentation process has on the sorting of ROI configurations.

3) Tour degradation for AD-P appeared in planning problems that replanned over 35% of the final tour. This was higher than expected as AD-P replans locally. The maximum tour degradation occurred at 84.9% replan effort for AD-P (*2x2m_9*) with a tour length 27.6% greater than AD-R. Despite the significant increase, AD-P still calculated the tours 1.5 times faster than AD-R.

4) The OSP was likely to produce smaller covering sets from the *redundancy-five roadmaps* than a random sampling procedure. As the initial *redundancy-ten roadmap*

was being removed, the *redundancy-five roadmap*, using the OSP, was unable to create covering sets that improved upon the quality of the initial plans.

5) Despite the *plan repair strategy* exhibiting tour degradation when replanning greater than 35%, the amount of computational effort expended by the *plan repair strategy* per unit of tour generated was significantly less than the *full replan strategy*. The RCE performance between the replanning strategies grew to as large as 58 times difference (*8x8m_1*) when *plan repair strategy* was able to preserve 98.5% of the existing tour. Even when the AD-P replanned over 85% of the existing tour, and exhibited tour degradation of upwards of 27%, the RCE between replanning strategies was 68% greater.

6) An examination between AD-P and AD-P(R) highlighted two limitations with the *plan repair strategy*. The preservation of the initial tour order and the sorting of configurations without any environmental analysis can potentially degrade tour quality. However, despite these limitations, the *plan repair strategy* did ensure that a feasible path could exist without replanning the entire tour with significantly shorter replan times.

8.4 Experiment 2: Adaptive Coverage Planning with Full Map Updates in the Representative Tank Environments

This experiment tests the ability of AD-P and AD-R to replan new coverage within the representative submarine tank environments *Tank-P1* to *Tank-P4* (Section 8.2.3). As performed in *Experiment 1*, one update occurs and each planning problem introduced a set of pipes(s), with *Tank-P1* corresponding to the introduction of one pipe and *Tank-P4* comprising of four pipes.

For this sequence of scenarios, Online CSP times were expected to grow due to the increasing size of the environment but Online MPP were expected to dominate replanning times due to the complexity of the planning problem, as seen from the results of the experiments conducted in *Chapters 4-6*. Given the complexity of the environments, the *hybrid-heuristic* was also tested to determine how effective was is on decreasing Online MPP times in an online context (Section 8.2.8). Therefore, this experiment tested the response of four *adaptive coverage planners*, AD-P, AD-R, AD-P(HY) and AD-R(HY) across the representative tank environments. Each planning problem was trialled 50 times. In total, 800 trials were conducted across all planning environments and *adaptive coverage*

planners.

8.4.1 Computational Observations and Results

Analysis of the Replanning Effort Required to Perform a Planning Update

Table 8-4 provides the statistical analysis of the results performed over all the trials. For AD-P and AD-P(HY) replanning ranged from 17.9% (*Tank-P1*) to 52.8% (*Tank-P4*). In line with expectations presented earlier in Section 8.2.3, this experiment demonstrated that AD-P easily accommodated a 52.8% replan effort and still outperformed AD-P(HY), AD-R and AD-R(HY) (Figure 8-28). AD-P solved the planning problem 96.7% to 99.6% faster than AD-R, 94.6% to 97.3% faster than AD-R(HY) and 53.8% to 56.0% faster than AD-P(HY) (Table 8-5). Consequently, all planning times for AD-P observed a statistically and practically significant result against the aforementioned *adaptive coverage planners* ($p < 0.001$, $d > 0.8$; Table 8-5). Unless stated otherwise, for all the planning attributes recorded for comparison between the four *adaptive coverage planners*, they all recorded a statistically and practically significant result ($p < 0.001$, $d > 0.8$; Table 8-5).

Introduction of the Hybrid-heuristic: Planning Times and Path Evaluations

The *hybrid-heuristic* significantly reduced planning times for AD-R(HY) compared to AD-R by up to 81.3%. As expected, the *hybrid-heuristic* reduced the number of path evaluations for AD-R(HY) compared to AD-R by up to 61%. However, despite these reductions, AD-R(HY) was not able solve any planning problem faster than AD-P or AD-P(HY). While AD-R(HY) and AD-P(HY) have similar costs to compute the *hybrid-heuristic* (Figure 8-29), the amount of extra replanning that AP-R(HY) was required to perform did not allow AD-R(HY) to outperform AD-P(HY). AD-R(HY) is required to replan between 1300 to 1700 more configurations than AD-P(HY) and AD-P.

Despite both AD-P(HY) and AD-R(HY) not being able to outperform AD-P, both *hybrid-heuristic* driven *adaptive coverage planners* do reduce the number of path evaluations required to solve the MPP in comparison to their respective Euclidean-based counterparts. Besides *Tank-P1*, which only recorded a 1.7% improvement ($p = 0.06$), AD-P(HY) evaluated 19.1% to 30.3% fewer paths than AD-P. AD-R(HY) observed a 49.4% to 61.1% reduction in the number of paths evaluated compared to AD-R. This reduction led to a 73.6% to 81.4% reduction in planning times compared to AD-R. For smaller replanning sizes AD-P(HY) evaluated up to 88% fewer paths than AD-R(HY). This value decreased to 66.5% as the replanning effort of AD-P(HY) increased to 57.2%.

Table 8-4: Statistical analysis between planning attributes of the AD-P and AD-R initialised with the *Euclidean assumption* and *hybrid-heuristic*.

Model / Planner / # Features			*Overall Planning Time (s)*		*ROI Validation (s)*		*Online CSP (s)*		*Online MPP (s)*		*Tour Length (m)*		*Configurations*				*Planning Iterations#*		*Path Evaluations#*		*Tour Replanned (%)**	*Tour time (mins)^*	*RCE*
			$\bar{x}$	*SD*	$\bar{x}$	*SD*	$\bar{x}$	*SD*	$\bar{x}$	*SD*	$\bar{x}$	*SD*	$\bar{x}$	*SD*	*New*	*Removed*	$\bar{x}$	*SD*	$\bar{x}$	*SD*			
Tank-P1	*E*	*P*	*3.37*	*0.07*	*1.00*	*0.01*	*2.23*	*0.06*	*0.13*	*0.04*	*149.06*	*0.49*	*1,418.00*	*4.11*	*99.00*	*7.10*	*39.74*	*4.39*	*280.40*	*12.69*	*17.92*	*72.11*	*242.92*
		R	*853.01*	*227.15*	*0.06*	*0.001*	*26.98*	*0.77*	*825.89*	*227.18*	*152.35*	*0.86*	*1,495.60*	*10.06*	*1,493.60*	*1,317.00*	*792.42*	*218.45*	*5,916.56*	*819.26*	*100.00*	*75.24*	-
	H	*P*	*69.97*	*0.90*	*1.00*	*0.01*	*2.23*	*0.05*	*62.31*	*0.60*	*149.01*	*0.43*	*1,417.66*	*3.68*	*98.68*	*7.24*	*38.92*	*5.11*	*275.72*	*12.31*	*17.90*	*72.09*	*2.18*
		R	*159.03*	*15.02*	*0.05*	*0.001*	*27.30*	*0.87*	*131.59*	*15.10*	*152.20*	*1.06*	*1,492.22*	*10.61*	*1,490.22*	*1,317.00*	*60.58*	*14.82*	*2,300.98*	*90.50*	*100.00*	*75.11*	-
Tank-P2	*E*	*P*	*11.50*	*1.68*	*1.38*	*0.01*	*8.21*	*0.15*	*1.88*	*1.69*	*155.60*	*0.67*	*1,497.06*	*5.81*	*178.06*	*15.60*	*86.84*	*45.36*	*616.76*	*76.79*	*25.94*	*75.84*	*70.88*
		R	*842.89*	*251.98*	*0.06*	*0.001*	*35.80*	*0.87*	*806.93*	*252.07*	*157.13*	*1.05*	*1,567.46*	*12.93*	*1,567.46*	*1,317.00*	*772.18*	*242.20*	*5,965.62*	*841.12*	*100.00*	*78.44*	-
	H	*P*	*77.96*	*0.82*	*1.38*	*0.01*	*8.22*	*0.16*	*63.47*	*0.59*	*155.46*	*0.63*	*1,495.74*	*4.51*	*176.74*	*15.60*	*34.74*	*6.13*	*473.26*	*27.46*	*25.87*	*75.77*	*2.20*
		R	*177.63*	*15.16*	*0.06*	*0.001*	*35.86*	*1.05*	*141.60*	*15.20*	*157.20*	*1.16*	*1,568.76*	*11.15*	*1,566.76*	*1,317.00*	*67.82*	*14.74*	*2,473.54*	*95.02*	*100.00*	*78.49*	-
Tank-P3	*E*	*P*	*14.60*	*1.50*	*1.72*	*0.02*	*11.07*	*0.26*	*1.76*	*1.54*	*165.91*	*0.70*	*1,597.80*	*7.16*	*278.80*	*27.56*	*131.02*	*45.47*	*894.90*	*75.06*	*40.69*	*80.91*	*52.02*
		R	*775.37*	*178.65*	*0.07*	*0.001*	*39.68*	*1.00*	*735.52*	*178.73*	*164.31*	*1.05*	*1,657.26*	*12.38*	*1,655.26*	*1,317.00*	*701.54*	*171.11*	*5,732.46*	*599.05*	*100.00*	*82.63*	-
	H	*P*	*82.91*	*0.94*	*1.72*	*0.02*	*11.13*	*0.22*	*64.85*	*0.63*	*165.84*	*0.70*	*1,596.22*	*5.99*	*277.22*	*28.72*	*83.66*	*7.82*	*765.28*	*30.64*	*40.63*	*80.85*	*2.20*
		R	*186.15*	*16.25*	*0.07*	*0.001*	*40.09*	*1.12*	*145.89*	*16.17*	*164.42*	*1.17*	*1,658.32*	*12.63*	*1,656.32*	*1,317.00*	*70.06*	*15.72*	*2,595.44*	*78.02*	*100.00*	*82.68*	-
Tank-P4	*E*	*P*	*24.11*	*2.95*	*2.56*	*0.02*	*18.44*	*0.42*	*3.06*	*2.93*	*174.48*	*1.14*	*1,659.26*	*7.29*	*340.26*	*44.26*	*205.80*	*79.85*	*1,213.56*	*110.13*	*52.72*	*84.39*	*30.78*
		R	*749.87*	*190.49*	*0.07*	*0.001*	*49.63*	*1.26*	*700.05*	*190.66*	*168.04*	*0.88*	*1,717.88*	*12.31*	*1,715.88*	*1,317.00*	*559.46*	*153.07*	*5,119.02*	*496.31*	*100.00*	*85.27*	-
	H	*P*	*91.22*	*0.85*	*2.55*	*0.02*	*18.47*	*0.32*	*64.49*	*0.56*	*174.68*	*1.62*	*1,660.48*	*7.17*	*341.48*	*42.78*	*120.72*	*9.17*	*1,019.06*	*30.49*	*52.76*	*84.46*	*2.14*
		R	*197.69*	*14.08*	*0.08*	*0.001*	*49.57*	*1.31*	*147.93*	*14.06*	*168.07*	*1.38*	*1,721.30*	*13.57*	*1,719.30*	*1,317.00*	*59.26*	*11.31*	*2,590.00*	*94.12*	*100.00*	*85.39*	-

ROI – Region of Interest *CSP – Coverage Sampling Problem* *MPP – Multi-goal Planning Problem* *P – AD-P* *R – AD-R* *E – Euclidean assumption* *H – Hybrid-heuristic*

Diff – Difference in the number of configurations in final tour compared to initial tour

** Number of configurations in final tour replanned. Excludes start and end positions*

*^Tour time = 2s*Configurations +0.1m/s*Tour Length*

Table 8-5: Relative performance and statistical significance between planning attributes between all *adaptive coverage planners*.

Model	*Overall Planning Times (s)*				*Configurations*				*Tour Length (m)*				*Path Evaluations*			
	RD	*PD %*	*p*	*d*	*RD*	*PD %*	*p*	*d*	*RD*	*PD %*	*p*	*d*	*RD*	*PD %*	*p*	*d*
AD-P vs AD-R																
Tank-P1	*0.004*	*-99.61*	*<0.001*	*7.48*	*0.95*	*-5.19*	*<0.001*	*10.95*	*0.98*	*-2.16*	*<0.001*	*4.87*	*0.05*	*-95.26*	*<0.001*	*13.55*
Tank-P2	*0.01*	*-98.64*	*<0.001*	*6.56*	*0.96*	*-4.49*	*<0.001*	*7.51*	*0.99*	*-0.97*	*<0.001*	*1.76*	*0.10*	*-89.66*	*<0.001*	*11.65*
Tank-P3	*0.02*	*-98.12*	*<0.001*	*8.45*	*0.96*	*-3.59*	*<0.001*	*6.09*	*1.01*	*0.97*	*<0.001*	*1.83*	*0.16*	*-84.39*	*<0.001*	*14.35*
Tank-P4	*0.03*	*-96.76*	*<0.001*	*7.50*	*0.95*	*-3.51*	*<0.001*	*5.98*	*1.04*	*3.83*	*<0.001*	*6.37*	*0.24*	*-76.29*	*<0.001*	*12.88*
AD-P(HY) vs AD-R(HY)																
Tank-P1	*0.44*	*-56.00*	*<0.001*	*11.18*	*0.95*	*-5.00*	*<0.001*	*10.43*	*0.98*	*-2.10*	*<0.001*	*4.28*	*0.12*	*-88.02*	*<0.001*	*39.40*
Tank-P2	*0.40*	*-56.11*	*<0.001*	*12.48*	*0.96*	*-4.65*	*<0.001*	*9.33*	*0.99*	*-1.10*	*<0.001*	*1.93*	*0.19*	*-80.87*	*<0.001*	*32.66*
Tank-P3	*0.46*	*-55.46*	*<0.001*	*12.01*	*0.96*	*-3.74*	*<0.001*	*6.67*	*1.01*	*0.86*	*<0.001*	*1.52*	*0.29*	*-70.51*	*<0.001*	*33.69*
Tank-P4	*0.46*	*-53.86*	*<0.001*	*14.26*	*0.96*	*-3.53*	*<0.001*	*5.86*	*1.04*	*3.94*	*<0.001*	*4.41*	*0.39*	*-60.65*	*<0.001*	*25.21*
AD-P vs AD-P(HY)																
Tank-P1	*0.05*	*-95.19*	*<0.001*	*136.78*	*1.001*	*0.02*	*0.66*	*-*	*1.0003*	*0.03*	*0.60*	*-*	*1.02*	*1.70*	*0.06*	*-*
Tank-P2	*0.15*	*-85.25*	*<0.001*	*53.21*	*1.009*	*0.09*	*0.21*	*-*	*1.0009*	*0.09*	*0.30*	*-*	*1.30*	*30.32*	*<0.001*	*2.75*
Tank-P3	*0.18*	*-82.40*	*<0.001*	*56.07*	*1.001*	*0.10*	*0.23*	*-*	*1.0004*	*0.04*	*0.64*	*-*	*1.17*	*16.94*	*<0.001*	*2.45*
Tank-P4	*0.26*	*-73.57*	*<0.001*	*35.28*	*0.993*	*-0.07*	*0.40*	*-*	*0.9988*	*-0.12*	*0.47*	*-*	*1.19*	*19.09*	*<0.001*	*2.77*
AD-R(HY) vs AD-R																
Tank-P1	*0.19*	*-81.36*	*<0.001*	*5.73*	*0.998*	*-0.23*	*0.11*	*-*	*0.999*	*-0.09*	*0.46*	*-*	*0.39*	*-61.11*	*<0.001*	*7.95*
Tank-P2	*0.21*	*-78.93*	*<0.001*	*4.98*	*1.001*	*0.08*	*0.59*	*-*	*1.001*	*0.05*	*0.74*	*-*	*0.41*	*-58.54*	*<0.001*	*7.46*
Tank-P3	*0.24*	*-75.99*	*<0.001*	*6.05*	*1.001*	*0.06*	*0.67*	*-*	*1.001*	*0.07*	*0.62*	*-*	*0.45*	*-54.72*	*<0.001*	*9.27*
Tank-P4	*0.26*	*-73.64*	*<0.001*	*5.40*	*1.002*	*0.19*	*0.19*	*-*	*1.001*	*0.02*	*0.90*	*-*	*0.51*	*-49.40*	*<0.001*	*8.57*
AD-P vs AD-R(HY)																
Tank-P1	*0.02*	*-97.88*	*<0.001*	*20.63*	*0.95*	*-4.97*	*<0.001*	*10.08*	*0.98*	*-2.07*	*<0.001*	*4.06*	*0.12*	*-87.81*	*<0.001*	*39.16*
Tank-P2	*0.07*	*-93.53*	*<0.001*	*19.73*	*0.95*	*-4.57*	*<0.001*	*8.46*	*0.99*	*-1.02*	*<0.001*	*1.74*	*0.25*	*-87.81*	*<0.001*	*21.62*
Tank-P3	*0.08*	*-92.16*	*<0.001*	*19.34*	*0.96*	*-3.65*	*<0.001*	*6.12*	*1.01*	*0.90*	*<0.001*	*1.59*	*0.34*	*-65.52*	*<0.001*	*22.22*
Tank-P4	*0.12*	*-87.80*	*<0.001*	*20.38*	*0.96*	*-3.60*	*<0.001*	*5.95*	*1.04*	*3.81*	*<0.001*	*5.09*	*0.47*	*-53.14*	*<0.001*	*13.48*

RP - Average of relative performance between each trial s.t. mean(AD-P/Compare) *p – Independent-samples t-test ($\alpha < 0.01$)*
*PD - Average of percentage differences between each trial s.t. mean((AD-P - Compare)/ Compare)*100* *d - Cohen's d effect size*

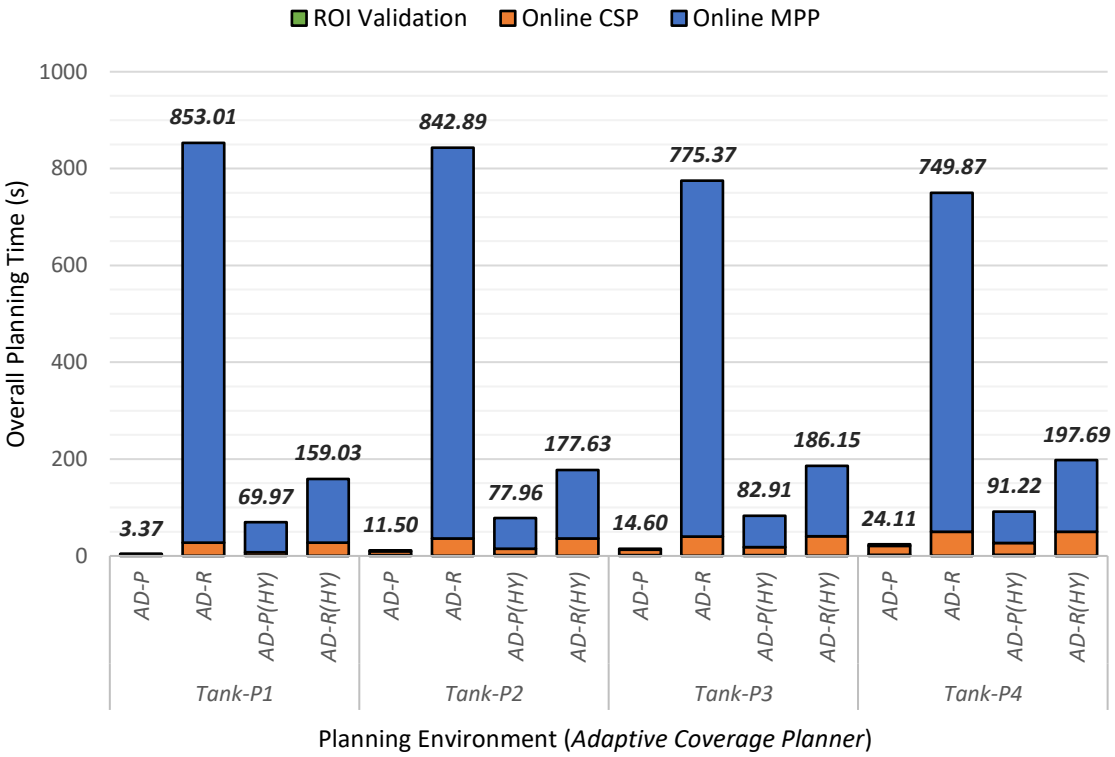


Figure 8-28: Overall planning times of each *adaptive coverage planner* over each representative tank environment.

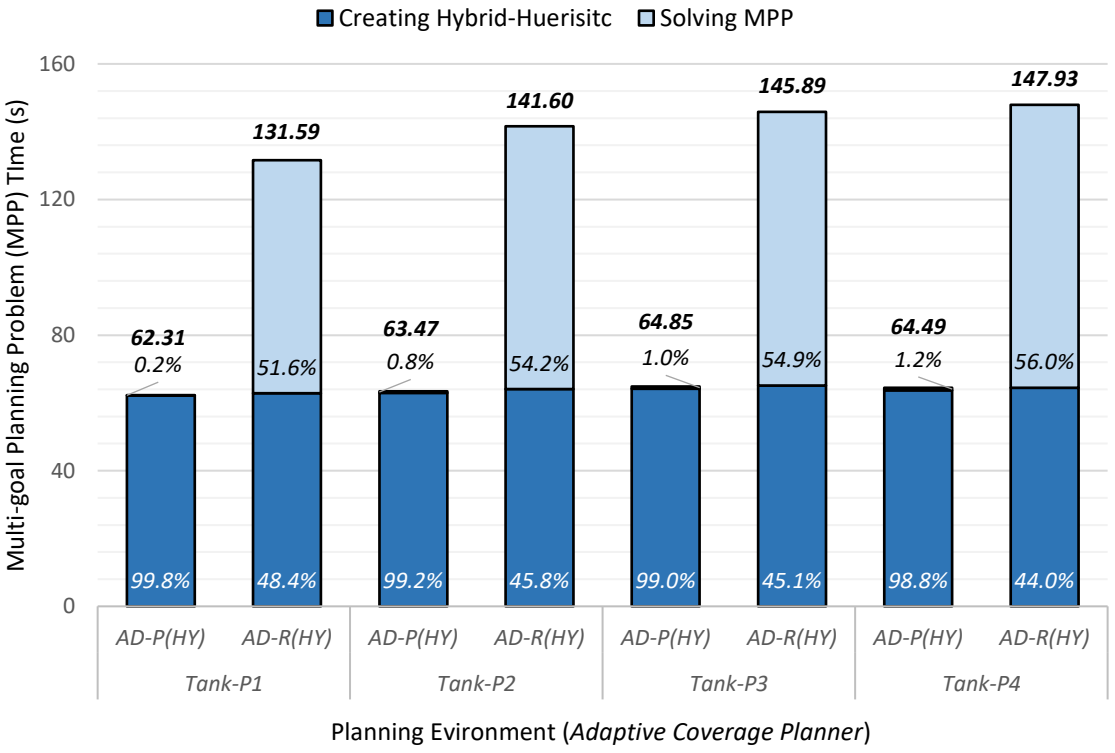


Figure 8-29: The time each Online MPP for AD-P(HY) and AD-R(HY) spent creating the *hybrid-heuristic* and solving all *multi-goal planning problem*s.

Despite AD-P(HY) evaluating significant fewer paths, the calculation of the *hybrid-heuristic* inhibits AD-P(HY) being the better *adaptive coverage planner*. The analysis of the results shows that overall planning times for AD-P(HY) would have been faster than AD-P if the calculation of the *hybrid-heuristic* was negligible. For *Tank-P4*, the calculation of the *hybrid-heuristic* accounts for over 98.8% of the Online MPP time and 99.8% for *Tank-P1* (Figure 8-29). If the time to compute the *hybrid-heuristic* was removed from the Online MPP time, AD-P(HY), for *Tank-P4*, solved the MPP (0.77s) 3.9 times faster than AD-P (3.05s). In summary, although the *hybrid-heuristic* reduces the number of path evaluations the LPP is required to evaluate to solve a given MPP, the additional time to calculate the *hybrid-heuristic*, over these planning environments at least, has outweighed the benefit of solving the MPP faster.

Distribution of Overall Planning Times among Planning Processes

Figure 8-30 shows the distribution of overall planning times across each of the planning processes as a ratio of the overall planning times for each *adaptive coverage planner*. Unexpectedly, the majority of the processing time for AD-P was spent sampling. It was not expected to observe that the Online MPP would only contribute as much as 15% to the overall planning time given the MPP times of *Tank* and *Tank-P4* in previous chapters.

ROI Validation and Online CSP times were equivalent for AD-P(HY) and AD-P given that the same ROIs were being generated, consequently producing similarly sized covering sets ($p > 0.01$). However, as previously discussed, the calculation of the *hybrid-heuristic* resulted in the Online MPP dominating planning times, despite solving all the sub-MPPs faster than AD-P. As expected, Online MPP times dominated AD-R and AD-R(HY) planning times due to the size and complexity of the problem to be replanned.

As replanning within the representative tank environments were more complex in comparison to the controlled environments, the sampling and path planning processes take longer to compute. The *segment* and *merge* processes of the *plan repair strategy* was shown to have little impact on overall planning times. The results of AD-P and AD-P(HY) demonstrated that the processes that create the ROIs were suitable in an online scenario considering the significant impact the ROIs had on reducing the planning times of the original CSP and MPP procedures.

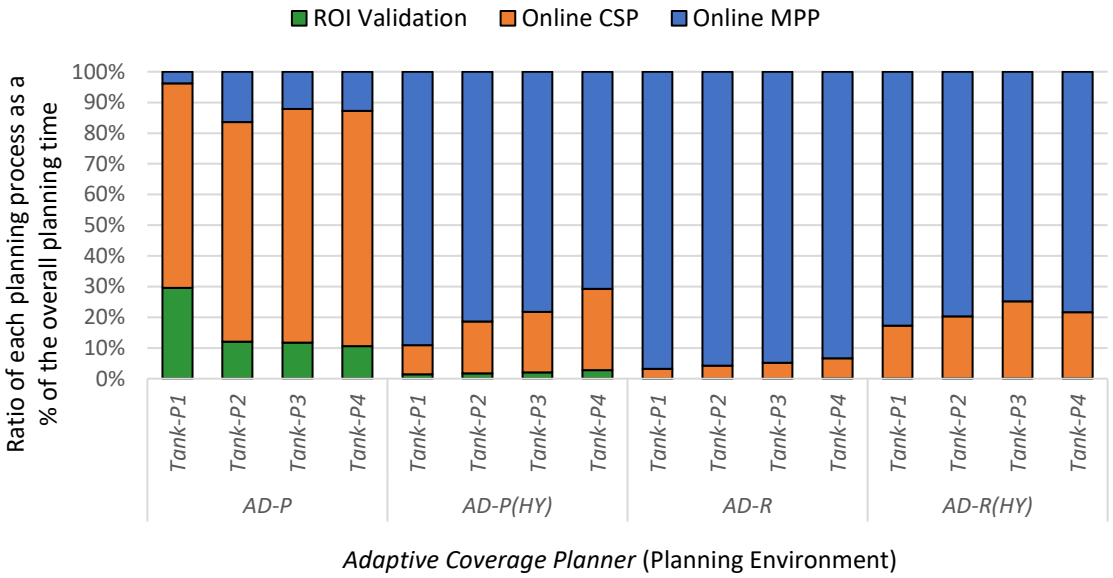


Figure 8-30: Ratio of each planning process for each *adaptive coverage planner* in the representative tank environments as a percentage of the overall planning time.

Differences in Covering Set Sizes.

There is a small difference in covering set sizes between AD-P and AD-R. The use of the OSP at a lower redundancy than the initial tour has increased the size of the covering set sizes of AD-R compared to AD-P by up to 5.2% (*Tank-P1*). This rate decreases to 3.1% the more replanning the *plan repair strategy* was required to perform. Again, this was due to the *plan repair strategy's* ability to preserve more configurations of the initial *redundancy-ten roadmap*. The results in Section 8.2.6 suggest that a *redundancy-ten roadmap* under a *random sampling procedure* would have yielded a smaller covering set size but at the expense of a longer Online CSP.

Planning Outcomes: Tour time, Tour Lengths and Relative Computational Effort

The *tour times* shows that despite the great disparity between planning times, there is only a two to three-minute difference in the time it would take to execute each of the tours. This is due to the 5% difference in the covering set size and 4.0% difference in tour lengths for any combination of the *adaptive coverage planner*.

Despite the considerable computational advantages the *plan repair strategy* possesses, the quality of the tour lengths did degrade when more replanning was required. Figure 8-31 shows how the tour lengths for both AD-P and AD-P(HY) degraded between *Tank-P2* and *Tank-P3*. This degradation corresponded to 23.9% and 40.7% of the final tours being replanned respectively.

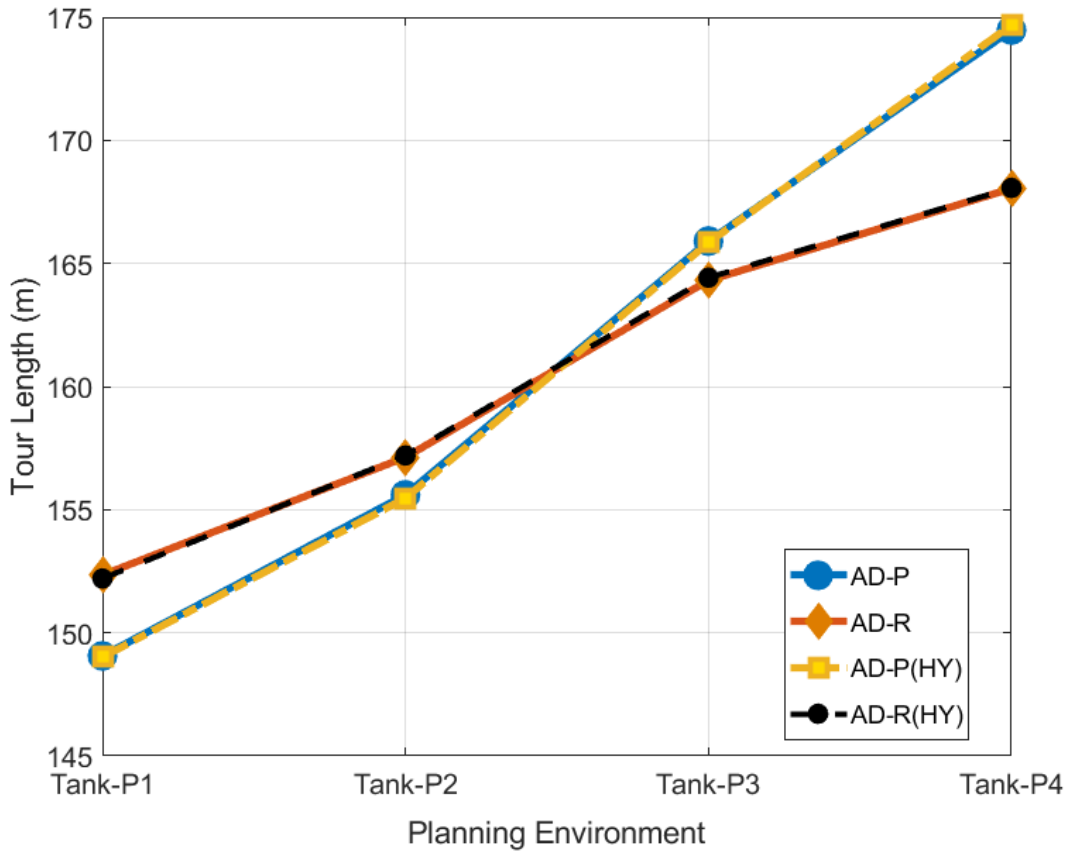


Figure 8-31: Tour length for each *adaptive coverage planner* for the representative tank environments.

The use of the *hybrid-heuristic* did not degrade tour length of AD-P(HY) and AD-R(HY) against their respective Euclidean counterparts. All tour lengths are within 1% for each test pair of *adaptive coverage planners*. There were occasions where AD-P(HY) produces tour lengths shorter than AD-P. However, these differences are negligible and could be to the slight variation in covering set sizes.

Despite the tour degradation and larger coverage set sizes exhibited by AD-P and to the lesser extent AD-P(HY), the RCE between each of the *adaptive coverage planner*, highlights the effectiveness of the *plan repair strategy* (Figure 8-32). The faster replanning times of AD-P again show how effective the *plan repair strategy* was when the replanning effort is small in comparison to the *full replan strategy* for all tested planning problems. As the replan effort increased, the RCE between AD-P and the other adaptive coverage planners decreased. However, the degradation of the tour quality exhibited by AD-P was arguably not enough to outweigh the computational benefits of the *plan repair strategy* over the *full replan strategy*.

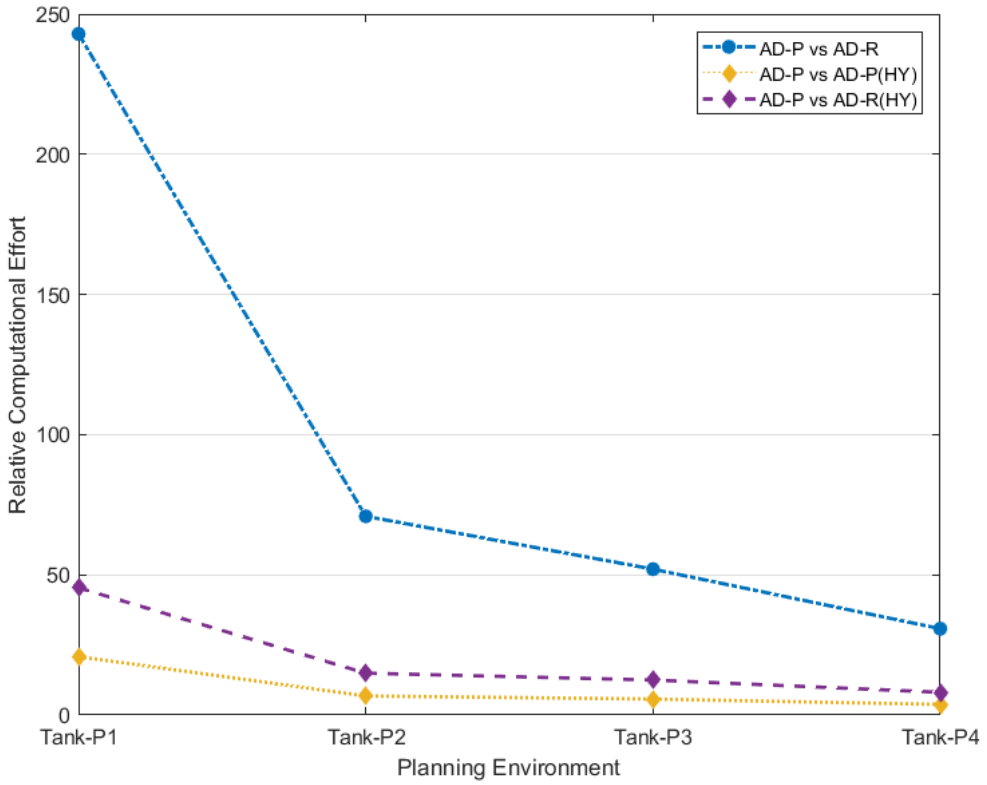


Figure 8-32: *Relative computational effort* of AD-P against AD-P(HY) AD-R and AD-R(HY).

8.4.2 Discussion

Given the result in *Experiment 1*, the results in this experiment follow similar trends which indicate that the AD-P is the superior of the two replanning strategies in this context. The *plan repair strategy* consistently outperformed both AD-R and AD-R(HY) as well as performing better than AD-P(HY). While *tour times* highlight that executing the resultant tours would take approximately the same amount of time to complete, the RCE, as shown in Figure 8-32, illustrates how much less work the *plan repair strategy* performs to produce a planning update.

The results of this experiment highlight the significant influence the ROI Validation phase has on reducing the amount of effort required by the sampling and path planning processes to perform a replanning update. Considering the size and complexity of the representative tank environments, the results illustrate that under the maximum replan effort, the *segment* and *merge* processes did not contribute significantly to the overall planning times for AD-P but have a positive impact at reducing the planning times of both the Online CSP and Online MPP processes.

It was expected that solving the Online MPP would dominate AD-P's planning times.

However, as the results show, Online MPP times were minimal as the Online CSP process took the majority of the processing time. This result was not observed for the AD-R and AD-R(HY) as the large covering set sizes, as shown throughout this thesis, resulted in the Online MPP dominating planning times. Given the main difference between the *plan repair strategy* and *full replan strategy* is how ROI Validation is used. The additional but negligible time it takes to generate ROIs to segment the plan has significantly reduced replanning times, making it feasible for the sampling and path planning processes of the *offline sampling-based coverage planner* to be used in an online context.

Despite the complexity of the representative tank environments used in this experiment, the limit to which the *full replan strategy* would outperform the *plan repair strategy*, using the RCE metric, was not found. While tour degradation occurred for AD-P and AD-P(HY) beyond a replan effort of 25.9% (*Tank-P2*), the degradation thereafter was not enough to outweigh the computational benefits of the *plan repair strategy.* Even at a maximum effort of 57.2%, the RCE between AD-P and AD-R was 30.8 times.

Even though a limit could not be accurately determined, it does appear that tour degradation will occur when around 30% of the final tour is required to be replanned. In *Experiment 1*, tour degradation occurred after 32.5%. In this experiment, tour lengths degraded after 25.9%. Tour degradation could be due to the limitations found in *Experiment 1* (Section 8.3.2). Despite the tour degradation observed in these results it was not enough to degrade the overall quality of the solutions provided by the *plan repair strategy*.

While the *additional termination conditions* have again been proven to provide stable LPP solutions, the addition of the *hybrid-heuristic* did not aid the *plan repair strategy* to produce solutions faster than AD-P. The *plan repair strategy* solved under the *Euclidean assumption*, in this planning context, is noticeably faster due to smaller replanning sizes and the overheads associated with calculating the *hybrid-heuristic*. The results showed that while the *hybrid-heuristic* did solve all sub-MPPs faster than the AD-P, as fewer planning iterations and path evaluations are executed, the computation of the heuristic outweighed any advantage to the LPP, limiting the AD-P(HY) to perform effectively.

These findings do not make the *hybrid-heuristic* completely unsuitable to be used in tandem with the *plan repair strategy*. If higher-fidelity constraints were placed on the planning problem, Online MPP times would increase. The use of the *hybrid-heuristic* would be suitable in this context as the overhead costs to compute the heuristic would be compensated

given the number of path evaluations that would no longer need to be solved. However, within this planning context, the *hybrid-heuristic* invalidates its use as a *suitable heuristic* as its computation outweighs the benefits of solving the MPP faster (Section 6.3, Criterion 3).

Given all the results shown over the previous two experiments, it was concluded that the *full replan strategy* is not suitable to convert the *offline sampling-based coverage planner* to the online domain. The results of the first two experiments have had both replanning strategies perform at the maximum replan effort over one planning iteration. On all accounts, the *plan repair strategy*, even with the *hybrid-heuristic* considerably outperformed the *full replan strategy*. AD-R, regardless of the size of the changes, will always be required to replan more than AD-P, unless the entire tour requires replanning. In that case a *full replan strategy* is better suited, given the additional *segment* and *merge* processes of ROI Validation and Online MPP.

Given these results, it is concluded that the performance between AD-P and AD-R will only continue to become more significant in a continually evolving environment. Furthermore, if higher-fidelity visibility and motion constraints were applied, again the performance between AD-P and AD-R would grow further apart, due to the more samples and paths AD-R would be required to replan. As it was expected that in a real-world scenario only a small amount of the environment is expected to change, 30% at most (*Assumption 3*), the designed *plan repair strategy* has proven to work sufficiently well when 52.7% of the existing tour is required to be replanned.

8.4.3 Summary

Expanding upon the findings of *Experiment 1*, *Experiment 2* has demonstrated the following:

1) The ROI Validation phase has significantly reduced the replanning effort for AD-P when planning in complex planning environments. The reduction led to AD-P outperforming the *full replan strategy* over all planning problems.
2) The *segment* and *merge* processes of ROI Validation and Online MPP have had minimal influences on computational times. As a result, the marginal time it does take to segment a tour results in a significant reduction in the number of configurations that need to be sampled and paths that will be required to be evaluated.
3) Despite the *hybrid-heuristic* solving the MPP faster than solving MPP using the

Euclidean assumption, the calculation of the heuristic outweighed the benefits when utilised in conjunction with the *plan repair strategy*.

Given these findings, it was concluded that the *full replan strategy* would not be suitable for online implementation since the degradation of the replanning strategy was significantly greater, the less replanning that AD-P was required to perform. There was a 202 times difference in RCE between AD-P and AD-R for *Tank-P1*.

8.5 Experiment 3: Simulated Coverage Planning with Partial Map Updates

The final experiment in this chapter simulates the motion of the robot conducting a live inspection plan. As discussed in Section 8.2.4, a mock mapping system was used to simulate the *mapping system module* of the *Inspection Planning Framework* (IPF) for a simulated robot implementing the simplified 6-DOF motion model used throughout all experiments thus far in this thesis. Given the analysis performed in *Experiments 1* and *2*, *Experiment 3* served as a validation of those findings but under more realistic conditions. This experiment aimed to determine whether if AD-P was capable of stratifying the final STIPP criteria by replanning under the one to two-minutes.

As *Experiment 1* showed how AD-R and AD-P respectively scaled to the size of the environment or the size of changes, only a subset of planning environments were used to determine whether AD-P was capable of outperforming AD-R in an iterative replanning situation. Table 8-6 shows all the planning environments and the number of trials undertaken in this experiment. An independent-samples t-test ($\alpha = 0.01$) proved statistically that enough trials had been conducted to demonstrate a difference between replanning strategies. In total, 505 trials were conducted across all planning environment and *adaptive coverage planners*. For completeness, no time limit was specified for *adaptive coverage planners* to complete the online planning problems.

From the findings of *Experiment 1*, AD-P was found to be insensitive to the offline tours. Consequently, in this experiment each planning problem was only initialised with the random tours (Section 8.2.5). The limitations discovered about the *plan repair strategy* in *Experiment 1* were not resolved for this experiment. AD-P was implemented as designed and the limitations are a subject for future work as discussed in *Chapter 9.*

Given the results of *Experiment 2*, AD-R(HY) was the only feasible hybrid method

Table 8-6: The number of trials for each planning problem conducted for *Experiment 3*.

Planning Environment	Variations	# of trials completed per variation per planner	
2x2m	{*_1, _5, _9*}	25	
4x4m	{*_1, _5, _9*}	25	
6x6m	{*_1, _5, _9*}	20	
8x8m	{*_1, _5, _9*}	10	
Tank	{*-P4*}	AD-P	20
		AD-R	5
	Total trials	505	

capability improving upon the Euclidean based method in the context of solving *Tank*. However, due to computational and memory limitations in creating the new STL files to enable skeletisation, the mapping system did not run in a reasonable amount of time. Consequently, AD-R(HY) was removed from this experiment.

8.5.1 Computational Observations and Results

Analysis of Overall Planning Times in Controlled Environments

The statistical analysis of *Experiment 3* is presented in Table 8-7. For the purposes of this analysis, the focus was only on the time taken by the *adaptive coverage planners* to supply inspection plan updates. Figure 8-33 illustrates the planning times together with the *average replan time per iteration* (ARTIP) for each planning problem. The planning times demonstrate AD-P replans each planning problem faster than AD-R on all accounts. The scale between planning times between AD-P and AD-R over all controlled planning environments is shown in Figure 8-34. For the smallest planning problem *2x2m_1*, AD-P replanned the planning problem *eight times faster than AD-R*. For the largest controlled planning environment *8x8m_9*, the speed-up between AD-P and AD-R increased to 32.8 times.

Outcome of Replanning Strategies in Representative Tank Environment

For the most complex environment tested, *Tank-P4, AD-P solves it 379 times faster than AD-R*. On average, AD-R took 46 hours to complete the replanning phase for the inspection over 367 replanning iterations; hence, the reason why five trials were sufficient to conclude that AD-R was not going to improve in future trials. The difference between AD-P and AD-R in this experiment demonstrated the ability of the new *plan repair strategy* to reduce processing times to replan in large and complex environments. Consequently, all planning

Table 8-7: Statistical analysis for *Experiment 3*. Shaded *tour times* indicate the robot will wait for tour updates.

Model / # Features		Planner /	# of replanning iterations		Overall Planning Time (s)		ROI Validation (s)		Online CSP (s)		Online MPP (m)		Configurations		Tour Length (m)		Path Evals ($x10^3$)		Tour Replanned (%)*	ARTPI (s)	Tour time (mins)^	RCE
			$\bar{x}$	SD	$\bar{x}$	SD	$\bar{x}$	SD	$\bar{x}$	SD	$\bar{x}$	SD	$\bar{x}$	SD	$\bar{x}$	SD	$\bar{x}$	SD				
2x2m	1	P	70.08	9.49	12.97	1.78	11.18	1.47	0.98	0.23	0.79	0.72	165.60	2.08	21.28	0.32	1.09	0.16	7.14	0.19	9.07	5.97
		R	69.36	11.18	104.17	16.15	3.08	0.59	86.19	13.55	13.86	2.25	228.60	11.48	27.51	0.81	9.80	1.51	96.05	1.50	12.20	
	5	P	145.80	16.84	40.02	5.25	29.62	3.41	3.19	0.47	7.15	3.43	200.08	3.51	25.38	0.84	4.27	0.54	29.09	0.27	10.90	8.56
		R	202.08	14.11	509.69	49.58	13.06	1.44	356.21	27.65	137.18	25.90	313.68	14.40	34.54	1.48	3.58	2.71	99.36	2.52	16.21	
	9	P	215.72	22.43	70.88	7.44	52.54	5.15	6.14	0.46	12.08	3.23	228.36	5.46	28.72	1.32	8.48	0.925	44.07	0.33	12.40	8.28
		R	285.20	17.38	939.97	71.05	24.95	2.50	589.80	35.49	320.40	54.27	391.84	12.87	40.79	1.41	58.41	3.60	99.49	3.30	19.86	
4x4m	1	P	35.40	9.13	19.41	5.50	17.78	4.68	0.72	0.27	0.88	0.96	500.48	2.22	73.95	0.36	0.65	0.18	2.40	0.54	29.01	8.63
		R	37.00	8.16	198.98	57.05	8.14	2.51	144.18	45.29	45.12	11.96	602.92	16.63	86.14	1.38	13.69	3.82	89.88	5.35	34.45	
	5	P	208.76	11.95	120.77	8.08	109.60	6.56	4.33	0.48	6.60	2.70	537.52	4.67	79.70	1.49	4.25	0.31	10.63	0.58	31.20	9.39
		R	228.20	18.03	$1.59x10^3$	109.88	49.61	6.14	$1.12x10^3$.	87.99	405.90	45.18	798.60	19.58	102.14	1.95	96.15	6.57	96.37	6.96	43.64	
	9	P	360.76	12.85	216.56	7.78	195.23	7.13	7.98	0.48	13.02	2.73	574.60	4.90	84.32	1.69	8.31	0.35	18.24	0.60	33.21	9.33
		R	422.88	19.63	12.97	1.78	104.48	7.87	0.98	0.23	0.79	0.72	966.08	27.01	115.60	2.65	175.12.	7.56	99.79	7.42	51.47	
6x6m	1	P	46.00	0.00	48.24	0.85	46.10	0.68	1.17	0.18	0.86	0.70	1,009.60	2.14	157.12	0.40	0.87	0.09	1.37	1.05	59.84	17.87
		R	38.85	8.05	982.08	274.75	13.34	3.36	559.29	150.85	405.43	132.37	1,155.85	27.12	177.88	2.76	3.9.60	10.02	93.16	25.12	68.17	
	5	P	247.55	14.38	265.55	15.55	251.69	14.51	6.02	0.51	7.27	2.38	1,047.45	4.83	161.57	1.56	4.59	0.23	6.15	1.07	61.84	15.60
		R	229.85	22.57	$5.60x10^3$	588.44	102.10	12.49	$3.14x10^3$	362.30	$2.34x10^3$	258.92	1,482.15	23.75	205.47	1.75	210.00	21.75	96.29	24.41	83.65	
	9	P	481.20	22.15	520.86	26.14	492.63	23.03	11.79	0.96	15.61	3.23	1,090.15	5.88	167.13	2.93	9.58	0.46	10.76	1.08	64.19	14.08
		R	438.30	29.15	$10.85x10^3$	$1.20x10^3$	228.41	21.15	$6.01x10^3$	699.01	$4.58x10^3$	568.84	1,729.95	28.06	224.13	2.05	378.86	33.84	99.88	24.79	95.02	
8x8m	1	P	55.80	0.63	85.31	1.96	80.19	1.27	1.42	0.29	3.59	1.25	1,744.90	2.02	275.59	0.37	1.22	102.33	0.67	1.53	104.09	28.21
		R	40.10	7.61	$2.67x10^3$	850.95	20.10	5.44	$1.10x10^3$	329.70	$1.54x10^3$	525.36	1,931.40	27.83	305.99	2.09	0.74	19.98	86.95	65.72	115.38	
	5	P	275.10	11.32	417.43	14.30	393.26	13.87	7.54	0.58	16.09	5.20	1,783.60	4.03	280.47	1.20	5.63	233.53	3.31	1.52	106.20	27.58
		R	234.00	18.71	$14.53x10^3$	$1.97x10^3$	147.81	12.60	$6.17 x10^3$	855.60	$8.19x10^3$	$1.13x10^3$	2,323.00	34.04	339.71	2.78	0.38	44.39	95.05	61.94	134.05	
	9	P	476.60	26.23	736.38	45.59	687.17	39.41	13.07	0.87	34.85	9.02	1,816.70	4.97	287.36	3.26	9.85	593.40	5.80	1.54	108.45	24.13
		R	432.20	33.46	$24.18x10^3$	$3.82x10^3$	327.31	36.76	$10.47x10^3$	$1.46x10^3$	$13.33x10^3$	$2.40x10^3$	2,612.40	39.64	362.88	3.24	608.93	74.15	99.92	55.90	147.56	
Tank	P4	P	288.10	19.39	437.46	34.28	325.72	22.80	45.16	1.91	65.65	15.01	1,703.70	14.75	199.78	6.25	48.38	4.65	26.03	1.52	90.09	278.73
		R	367.20	19.27	$165.7x10^3$	$13.1x10^3$	107.06	7.04	$8.59x10^3$	732.37	$157.3x10^3$	$12.43x10^3$	2,508.00	68.77	234.17	4.55	1028.82	74.50	99.92	451.74	122.63	

ROI – Region of Interest CSP – Coverage Sampling Problem MPP – Multi-goal Planning Problem P – AD-P R – AD-R RCE – Relative Computational Effort ARTIP – Average Replan time per Iteration

** Tour replanned is calculated as a ratio of how many original configurations exist within the final tour. Excludes start and finish positions.* *^Tour time = 2s*Configurations + 0.1m/s*Tour Length*

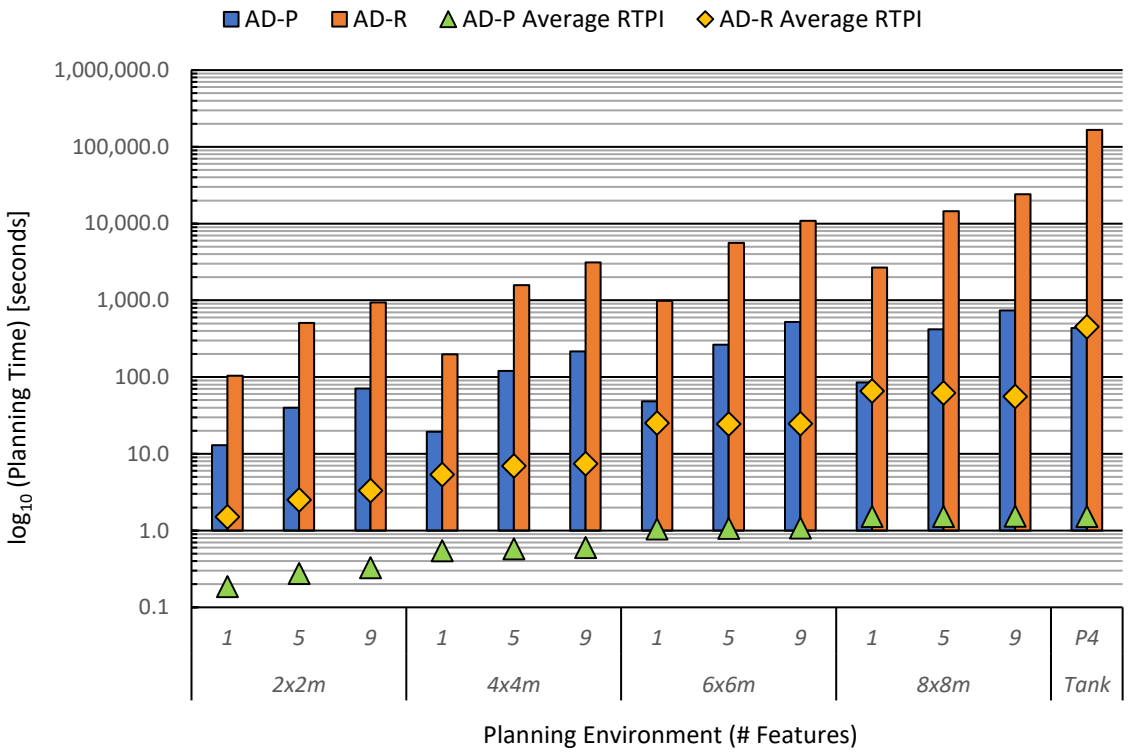


Figure 8-33: Overall planning times and the *average replan time per iteration* (ARTPI) for AD-P and AD-R over all tested environments.

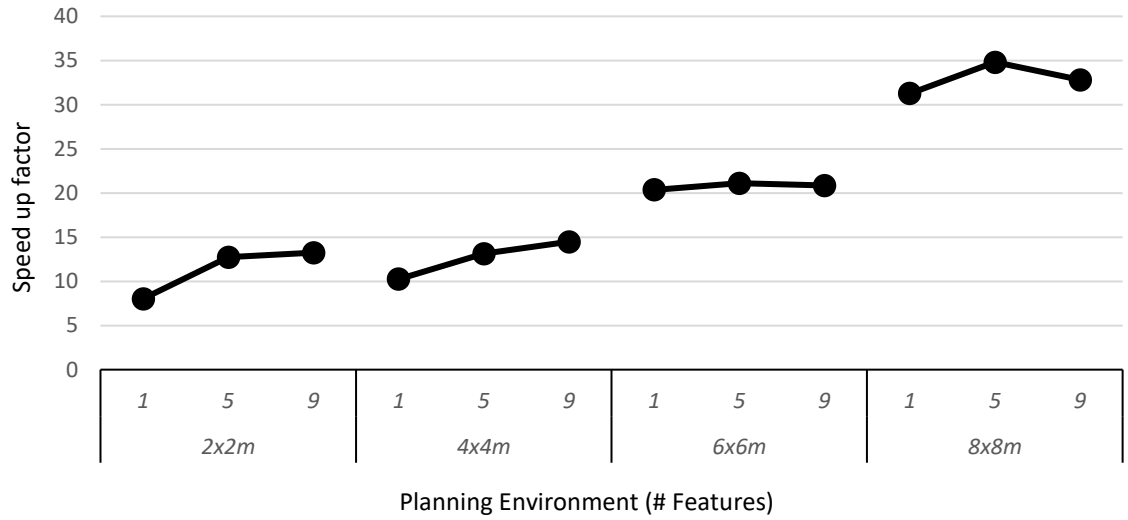


Figure 8-34: Speed-up of AD-P over AD-R for the controlled environments.

problems recorded a statistically and practically significant result ($p < 0.001$; $d > 0.8$); following the same trend in all trials tested throughout this chapter.

Average Replan Time per Iteration

As AD-P iteratively replanned smaller regions of the environment per replanning update, compared to AD-R, AD-P maintained consistent replanning times, taking no more than 1.6 seconds on average to provide a tour update over all tested environments. This performance compared to AD-R's ARTIP which steadily increased from seconds to minutes as the environments grew in size and complexity. For *Tank-P4*, AD-R's average RTIP was 7.5 minutes, equivalent to the total planning time taken for AD-P over the lifetime of the inspection; whose ARTIP was 1.5 seconds. Given that AD-R is equivalent to solving the inspection planning problem offline, an ARTIP of 7.5 minutes aligns with the 6.6-minute solution times ($\sigma = \pm 1.9$ minutes) *Tank-P4* was being solved in *Chapter 5* (Table 5-5). The differences between the offline and AD-R solutions can be attributed to;

1) the additional time to execute ROI Validation,
2) the difference in configuration locations due to OSP, and
3) the tour being solved between different start and finish locations.

Between these two ARTIP times, only AD-P meets the one to two replanning time set by *Requirement 7* of the STIPP criteria.

Distribution of Overall Planning Times among Planning Processes

Figure 8-35 and 8-36 show the distribution of the overall planning times amongst each of planning phases for AD-P and AD-R respectively. Due to AD-P's ability to replan smaller regions of the environment, the use of Online CSP and Online MPP procedures has decreased in comparison to what was observed in the previous two experiments. The ROI Validation phase accounts for up to 85% of the overall planning times, an average increase of up to 60% on what was observed in the previous experiments, when replanning was performed over a single iteration.

As expected, the planning environments with the higher replanning effort, *2x2m_(5,9)* and *Tank-P4*, required Online CSP and Online MPP to replan more thus reducing the influence of ROI Validation. However, the influence of Online CSP and Online MPP processes due to preservation, does not allow these processes to influence planning times as much as they do for AD-R. As ROI Validation in AD-R is only used to determine how much of the plan has

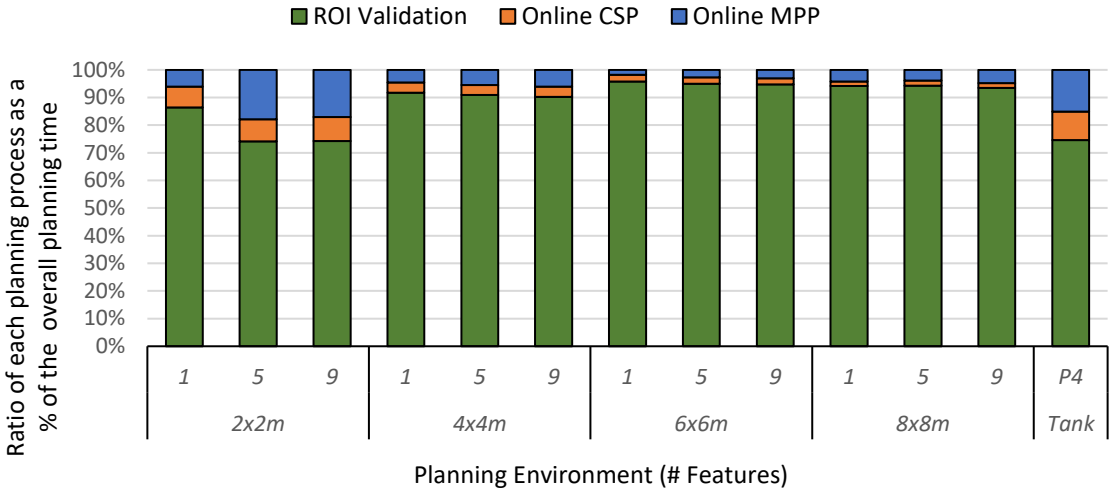


Figure 8-35: Distribution of each planning process as a ratio of overall planning time for AD-P.

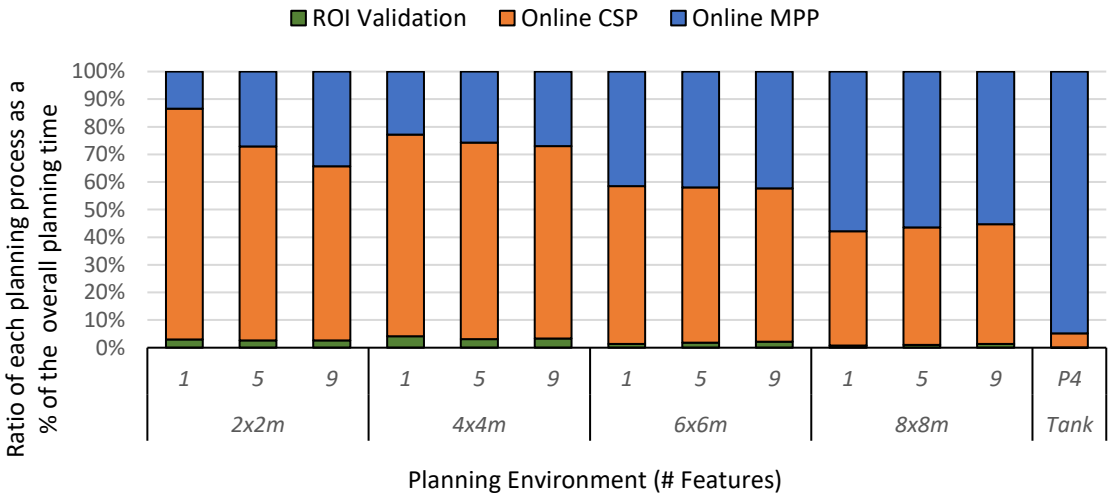


Figure 8-36: Distribution of each planning process as a ratio of overall planning time for AD-R.

been completed before the map update, the influence of ROI Validation is again minimal, as previously observed. As each replanning iteration is equivalent to solving the planning problem offline, it was no surprise that Online MPP accounted for approximately 95% of the overall planning times to solve *Tank-P4*; equivalent to the domination of MPP times observed in the offline experiments (Table 5-5).

Covering Set Sizes, Path Evaluations, Tour Lengths and Replan Effort

This experiment has shown that AD-P produced covering set sizes on average 27.5% smaller than AD-R over controlled environments ($p < 0.001$, $d > 0.8$) and 32% smaller for *Tank-P4* (Figure 8-37; $p < 0.001$, $d > 0.8$). Consequently, the larger covering set size of AD-R has led to a significant increase in the number of planning iterations and created longer tours than AD-P (Figure 8-37; $p < 0.001$, $d > 0.8$). For *Tank-P4*, AD-P produced tour lengths that

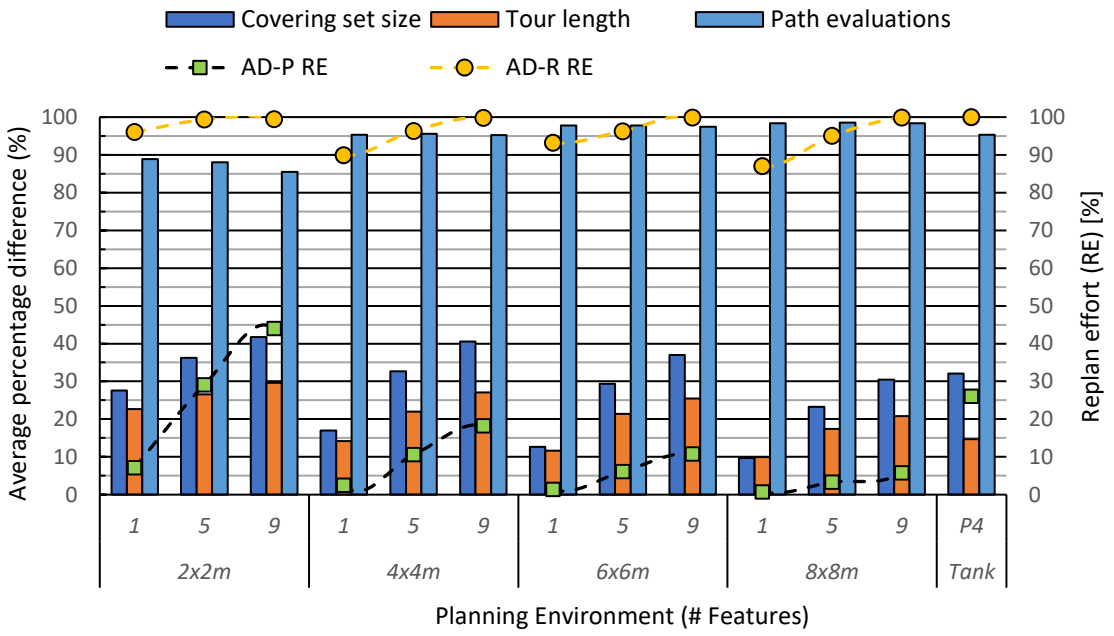


Figure 8-37: The additional increase in replanning led to AD-R producing covering set sizes, planning iterations and tour lengths that were larger than AD-P over all planning problems.

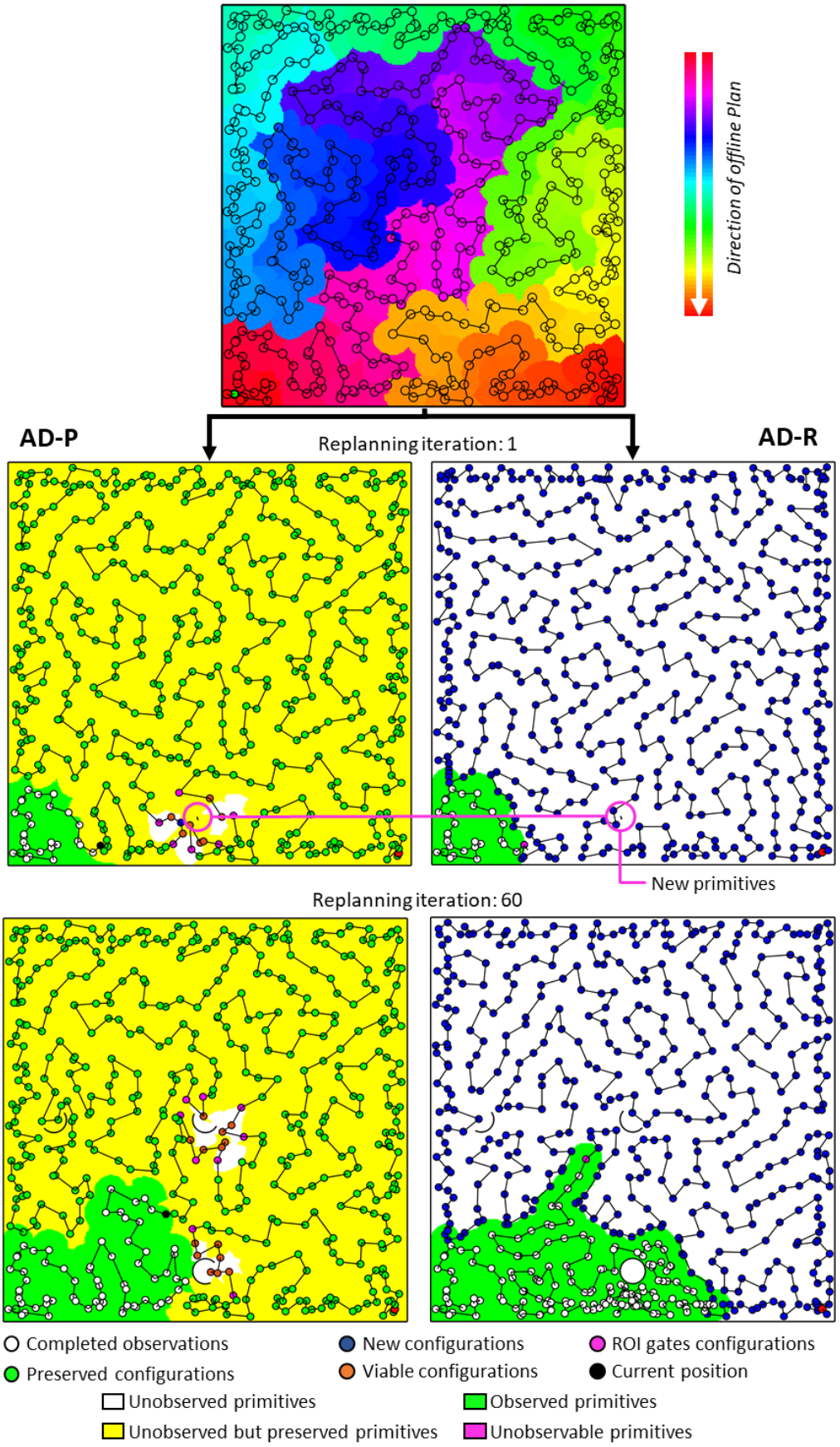
were 15% shorter than AD-R. This is a significant change considering that AD-P produced tours that were 5.2% longer than AD-R in *Experiment 2* (Table 8-5).

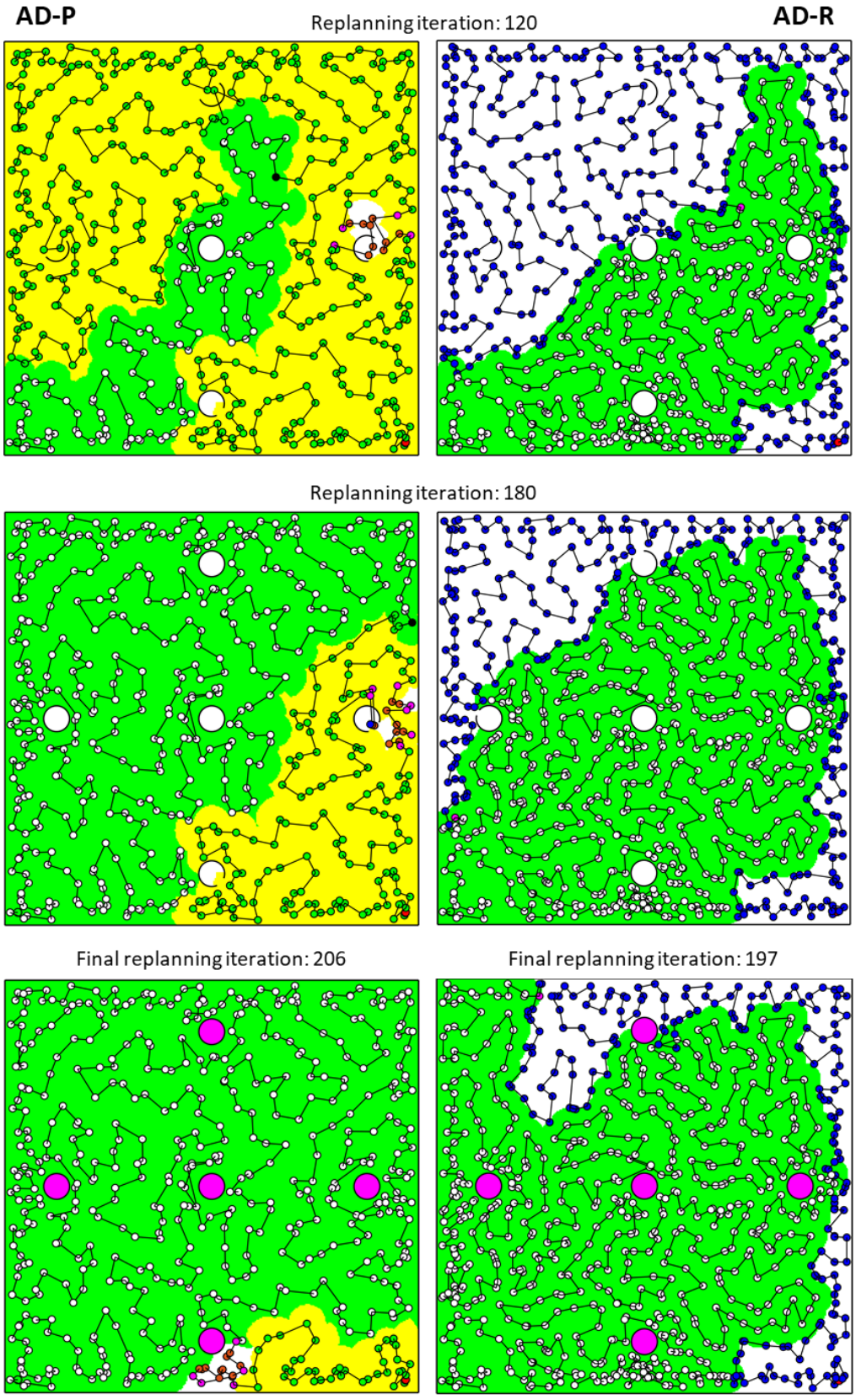
As the inspection plans were simulated, AD-R only had to replan everything again upon encountering the first map update. Therefore, for the planning environments that contained centrally located features, i.e *2x2m_(1)*, early segments of the inspection plan could be completed and thus preserved before replanning commenced. Therefore, the sooner the robot encountered change, the less AD-R was able to preserve. Consequently, AD-R showed a similar trend in replanning effort as AD-P (Figure 8-37).

Visual Demonstration of Replanning Strategies and Validation of Planning Heuristics

A visual comparison between a trial of AD-P and AD-R for *4x4m_5* is illustrated in Figure 8-38. In these trials, the regions completed by the robot are highlighted in green, the region preserved by AD-P displayed in yellow with the uncoloured regions representing the regions that required replanning. Over each replanning iteration presented, it is clear that;

1) AD-P replans less and AD-R,
2) the difference in covering set sizes,
3) the difference between tour lengths, and
4) that AD-P follows the strict ordering of the offline tour, only deviating due to the influence of new features.


Direction of offline Plan
AD-P
Replanning iteration: 1
AD-R
New primitives
Replanning iteration: 60
Completed observations
New configurations
ROI gates configurations
Preserved configurations
Viable configurations
Current position
Unobserved primitives
Observed primitives
Unobserved but preserved primitives
Unobservable primitives



Planner	Replan time (mins)	Configurations	Tour length (m)
AD-P	1.9	537	83.7
AD-R	23.4	801	103.2

Figure 8-38: Evolution of inspection plans for AD-P and AD-R for *4x4m_5*.

The trial also demonstrates the capability of the *Relaxed Trapped Configuration Heuristic* (R-TCH; Section 7.4.2), *Nearest Neighbour Heuristic* (NNH; Section 7.4.2) and *Trapped Configuration Heuristic for Path Planning* (TCH-PP; Section 7.5.2) by removing configurations that are contained within partial constructed structures. For the AD-P planning iterations, it can be seen that configurations exist within partially constructed features. In subsequent iterations, these configurations are removed as the features are completed in future iterations. While these functions are not the primary focus of the investigation, they were developed to ensure the *adaptive coverage planner* is capable of efficiently handling evolving environments. Furthermore, the trial highlights the *unobservable primitives* in the final planning iterations (magenta) to ensure the operator is aware of areas of the environment that cannot be inspected (*Requirement 4*).

Planning Outputs: Tour Time and Relative Computation Effort

The larger covering sets and longer tour lengths of AD-R solutions consequently resulted in longer *tour times* (Figure 8-39). Furthermore, the *tour times* degraded from the previous experiments, increasing from 13.7% to 28%, proportional to the size of the environment.

Planning problems whose *adaptive coverage planner's* overall replanning times are greater than the *tour time*, suggests that concurrent replanning would not be feasible, are shaded in Table 8-7. For AD-P, all replanning times were considerably under their respective *tour times*. This finding indicates, that for these planning problems, the *plan repair strategy* would have the capability of replanning concurrently, in accordance *Assumption 7* (Section 3.6.1). For AD-R, only the smaller planning problems were able to meet this requirement. Once the planning problem becomes larger in size, with more features, *tour times* and replanning times for AD-R show that the robot is certain to wait, for considerable amounts of time, for replanning updates.

Finally, due to AP-R's longer overall planning times and tour lengths, the RCE between the two replanning strategies considerably increased compared to the RCE of previous experiments. For the controlled planning environments, the RCE increases from 6 to 28 times the computational effort for equivalent tour quality. Granted, the RCE drops slightly when AD-P is subjected to replanning more of the environment however, the difference is negligible. For *Tank-P4*, the RCE between replanning strategies was 278 times greater, a significant increase compared to the 30.8 times difference observed in *Experiment 2* (Figure 8-32).

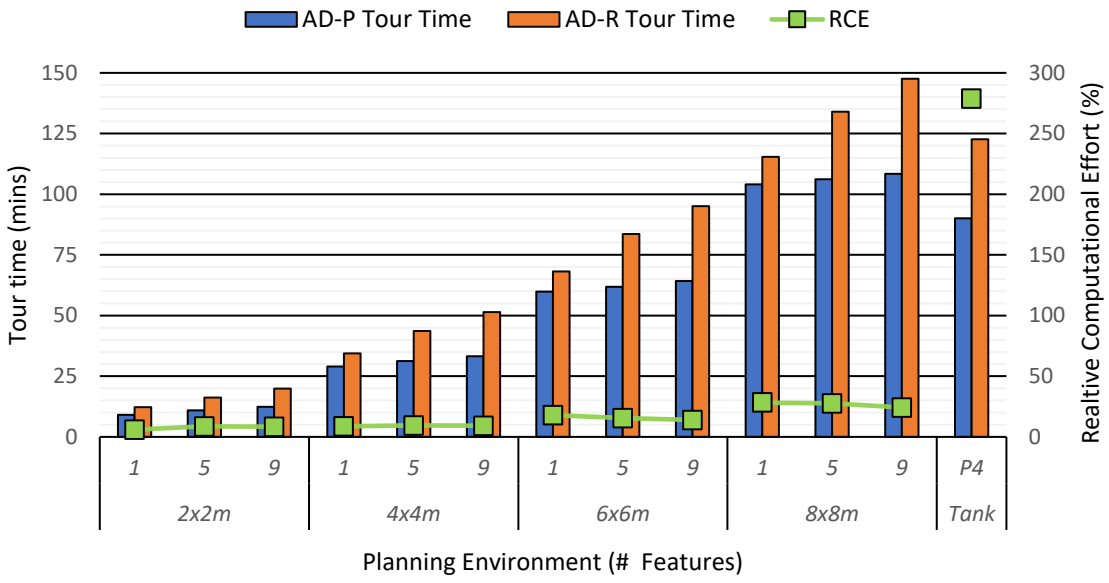


Figure 8-39: *Tour times* and *relative computation effort* (RCE) for *Experiment 3*.

8.5.2 Discussion

The analysis of this experiment clearly demonstrates how effective the *plan repair strategy* is at preserving uninfluenced segments of the tour to minimise the amount of replanning required to perform a tour update. For all tested cases, AD-P outperformed AD-R in all evaluated metrics. Furthermore, with AD-P producing smaller covering sets and shorter tour lengths compared to solutions generated by AD-R, the expected time to complete each AD-P tour would be faster than AD-R and the RCE demonstrates how efficient the *plan repair strategy* is at replanning.

This experiment served as a validation to the previous two experiments. Once again, the analysis highlights that the difference between the replanning strategies. With smaller amounts to replan per iteration, AD-P still scaled to the size of the changes as indicative of the overall planning times, while AD-R scales the size of the environment. Due to AD-P's ability to preserve the majority of the tour per replanning iteration, the difference between AD-P and AD-R's computational performance became significantly more distant the larger and more complex the planning scenarios became.

The results for *Tank-P4* demonstrate suitability of the *plan repair strategy* to solve a representation of the target environment. Even if the AD-R(HY) were to be used for this experiment, it would have only reduced Online MPP times. Online CSP would remain the same and considering that Online CSP times within the AD-R approach for *Tank-P4* were 20 times greater than AD-P's overall planning time.

While the trends of AD-P were replicated in this experiment, it was surprising to observe the significant tour degradation exhibited by AD-R solutions. In the previous experiments, when AD-P replanned more than 28 to 35% of the environment, tours lengths degraded in comparison to AD-R. However, in this experiment this did not occur for any of the planning problems, even when AD-P replanned 44% of the final tour for *2x2m_9*. A comparison between AD-R tour lengths also show how the tour lengths have degraded against the respective results in the previous experiments (Figure 8-40).

While it was not expected for either AD-P or AD-R to produce tours of better quality than the tour generated by their respective *Experiment 1* and *2* counterparts, they were able to reconsider the entirety of the environment before replanning, AD-R tour lengths were up as much as 72% longer. Only the larger planning problems with fewer features (*6x6m_1* and *8x8m_1*), which subsequently resulted in fewer replanning iterations, exhibited smaller differences in path lengths between the experiments. This finding would not be surprising if AD-P followed a similar behaviour. However, as illustrated in Figure 8-40, AD-P experiences minimal tour degradation and in the case of *2x2m_9*, produced shorter tours. It is likely for *2x2m_9* that the incremental nature of map updates did not result in the large ROI gates separation that was observed in *Experiment 1* (Figure 8-26).

As previously discussed, the tour degradation experienced by AD-R in the previous experiments was determined to be the result of using the OSP with a lower redundancy roadmap than was constructed for the initial tour. The increase in tour lengths in this experiment could be directly related to the increase in the covering set size AD-R generated over all planning problems (Figure 8-41). However, whilst the CSP is sensitive to the redundancy of the roadmap and the environment neither of these parameters have changed between the three experiments, therefore, suggesting that the significant increase in sub-optimality observed in this experiment by AD-R cannot be solely attributed to these parameters.

The only other differences between each experiment is the number of planning iterations performed and the resulting regions that require replanning. As AD-R solutions were clearly more impacted than AD-P solutions, the degradation cannot be directly related to the number of replanning iterations, as these are approximately the same for each planner but must be due to the size of the area that is required to be replanned.

While the data collected from this experiment could not conclusively determine the cause

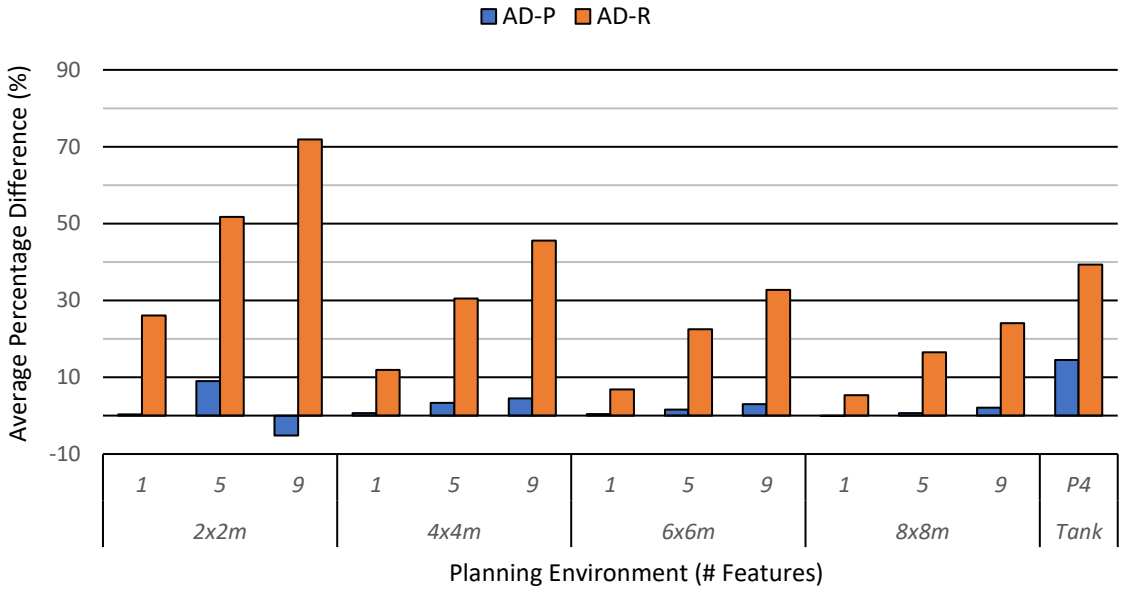


Figure 8-40: The average percentage difference between tour lengths produced in *Experiment 1* and *2* against the tour lengths produced in *Experiment 3*.

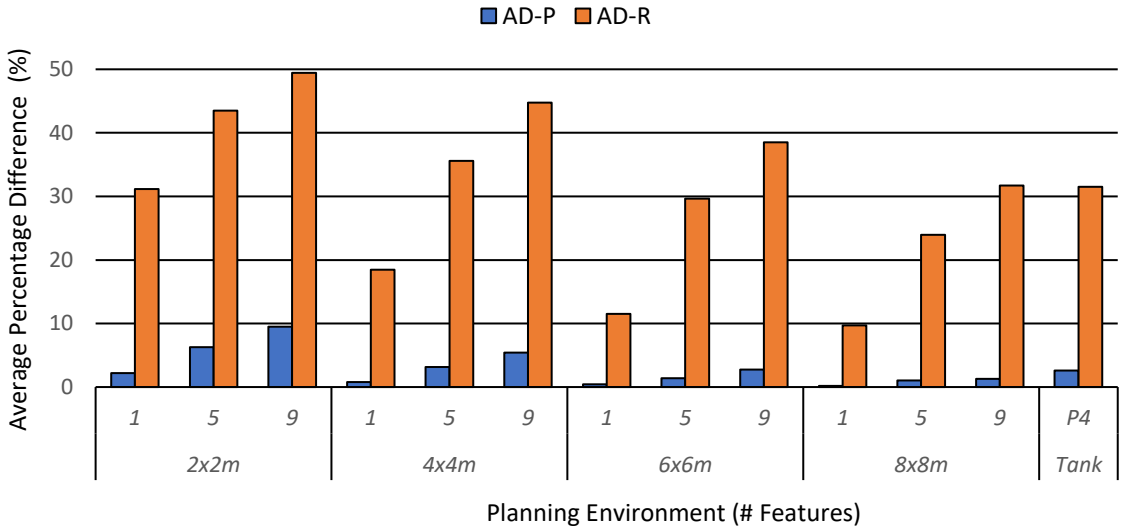


Figure 8-41: The average percentage difference between covering set sizes produced in *Experiment 1* and *2* against the covering set sizes produced in *Experiment 3*.

behind AD-R's tour degradation, a visual examination of the resultant tours allowed for a few possible reasons to explain the observed behaviour. For reference, these reasons are present in the *4x4m_5* trial shown in Figure 8-38 and listed below:

1) For both AD-P and AD-R, there is a clustering of configurations that appear on the boundaries that separated observed and unobserved primitives.

For AD-R this boundary exists between the executed and non-executed areas of the environment (green and white coloured primitives respectively). For AD-P this boundary

exists between the executed, preserved and unpreserved areas of the environment (green, yellow and white coloured primitives respectively). In this context, as the boundary for AD-R for *4x4m_5* were significantly larger than that of AD-P for any given replanning iteration, AD-R is more susceptible to boundary clustering. However, if larger areas were required to be replanned, resulting in a larger boundary defined by the ROIs, AD-P would have been more significantly impacted by boundary clustering. If this were the case, it may explain why boundary clustering was not present for AD-P for *Experiment 1* and *2*, since the no portion of the tour was executed, which resulted in better quality tours being presented with respect to AD-P despite using a lower redundancy roadmap as previously mentioned.

2) AD-R tours change direction several times over the course of multiple iterations.

It can be observed in Figure 8-38, the AD-R's tour changes several times while AD-P preserved the direction of the offline tour. Given that AD-R recalculates the entirety of the remaining tour based on the robot's current location and the amount of information present within the environment, as more information about the features becomes present, the direction of the tour may change in response to what is currently the *shortest* tour available. However, upon the next iteration when new information about the environment is delivered, resulting in a new set of configurations, the *shortest tour* to the goal may change again. As a result, over the course of the inspection the tour may change several times, increasing the overall tour length, making it significantly longer than a tour that is fixed to the direction of the offline tour when the level of replanning per iteration is significantly less.

The combination of these potential factors along with the removable of high-redundancy covering sets with lower redundancy roadmaps and the complexity of the environment at each replanning iteration all could compound together to result in the significant, yet unexpected tour degradation witnessed by AD-R.

To conclude, this experiment was also used to determine whether AD-P or AD-R could satisfy the one to two-minute replanning time requirement of the STIPP criteria (*Requirement 7*). As the analysis showed, all of AD-Ps ARTIP were able to successfully meet this requirement, with an ARTIP of 1.57s for *Tank-P4*. The ARTIP and overall planning times indicated that the robot would be able to concurrently replan and perform an inspection when utilising the *plan replan strategy*. While AD-R was able to replan under the one minute per iteration for the controlled environments, for *Tank-P4*, which was the

benchmark environment used, ARTIP was over 6.5 minutes, an undesirable outcome considering the results of AD-P.

The conclusion must be considered with respect to the visibility and motion constraints placed on the planning problems. The simplified constraints used create a situation where it is cheaper to replan than it would be with the actual high-fidelity constraints of a multi-legged platform. In this case, replanning times for both AD-P and AD-R would be greater than the times presented in these experiments. However, if high-fidelity constraints were used, it would greatly impact the *full replan strategy*. AD-P will continue to replan faster than AD-R until the replan effort is equivalent. Then, the additional cross-checks performed by ROI Validation would impact AD-P replanning times.

In the context of the STIPP and the *Inspection Planning Framework*, the constraints used for the *inspection planning module* throughout this thesis would have been implemented for physical testing on the concept demonstrator (Section 1.2.2). While the simplified constraints have had a significant impact on reducing planning times, especially MPP and Online MPP times, this thesis sought to reduce the impact high-fidelity constraints of a multi-legged platform would have on the *adaptive sampling-based coverage planner*. Therefore, the results of this simulated experiment provide a suitable indication of AD-P's performance. So, in conclusion, the results in *Experiment 3* validate that the *plan repair strategy* had successfully satisfied all the criteria of STIPP.

8.6 Chapter Conclusions and Summary

In this chapter, the *plan repair strategy* was examined over three experiments, to determine if it was the better of the two replanning strategies, using the *full replan strategy* as a baseline comparison, to adapt the *offline sampling-based coverage planner* to function adaptively in the online domain.

Experiment 1 analysed the intrinsic behaviour of both replanning strategies within the controlled environments that scaled in size and the number of features. As each replanning problem introduced each feature in full, it evaluated AD-P's capability to replan the maximum amount of change within a single replanning iteration. The results of *Experiment 1* clearly illustrated how AD-P's computation scaled with the size of the introduced features rather than the size of the environment. It also demonstrated that AD-P was capable of replanning multiple disconnected ROIs concurrently over any type of plan.

Experiment 2 tested AD-P over the representative tank environments in the same manner as *Experiment 1*; replanning occurred over one planning iteration. AD-P exhibited the same trends as seen in *Experiment 1*, where the computation of AD-P continued to scale with the size of the changes, unlike AD-R, whose planning times significantly increased due to the size and complexity of having to completely replan the entirety of the tank environments. Even with the assistance of the *hybrid-heuristic* and the tour degradation that was present above a replan effort of 28%, AD-R(HY) was not capable of outperforming either AD-P or AD-P(HY). The amount of replanning required within these environments was too great to compensate for a planning process that had the capability to segment and preserve the majority of the initial high-quality roadmaps.

It was clear from *Experiment 2* that AD-P was likely to outperform AD-R if both adaptive coverage planners had to perform iteratively over the lifetime of an inspection. This prediction was proven to be true in *Experiment 3*. *Experiment 3* served as a validation experiment to confirm that the trends and behaviours witnessed in the previous two experiments, were still valid in a simulated planning environment. This experiment tested both *adaptive coverage planners* under more realistic conditions given that real-world testing of the concept demonstrator was not able to be conducted at the time of this thesis.

The results in *Experiment 3* conclusively showed that AD-P was sufficiently capable at providing replanning solutions faster and more efficiently than AD-R. Tour lengths and configurations set sizes increased for AD-R due to creating lower redundancy roadmaps than was used to create the offline tour. With AD-P preserving more of the existing tour, AD-R could not match the quality of AD-P solutions as it continually replanned the entire environment upon each iteration.

The purpose of these experiments was to address a series of questions that would help determine which of the two replanning strategies was best suited to adapt the *offline sampling-based coverage planner* to the online domain. The answers to these questions, given the experimental results and analysis, are summarised as follows:

Question 1) What is the computational impact of segmentation and merging of sub-plans on the planning times?

The segmentation and merging processes of the *plan repair strategy* had negligible effects on the computational time when compared to the *full replan strategy*. While the ROI

Validation phase was the most dominate process for AD-P in *Experiment 3*, the effect that the segmentation process had on reducing Online CSP and Online MPP was significant. The segmentation process created smaller planning problems that led to smaller covering sets and MPPs. Consequently, these smaller problems solved quickly compared to a *full replan strategy*. Therefore, to conclude, the segmentation and merging processes of the *plan repair strategy* were not found to negatively impact planning times as originally expected. Of the time that was spent segmenting the tour per replanning iteration, it had a positive influence on planning times by successfully reducing the time spent using the sampling and path planning processes.

Question 2) What is the potential drop in tour optimality as the sub-plan approach does not solve the coverage plan globally?

Tour degradation was an expected feature of the *plan repair strategy*. However, tour degradation was only observed in the experiments that required a complete replan within one iteration. When the *plan repair strategy* was tested under maximal replanning loads in *Experiment 1* and *2*, tour degradation did occur when 35% and 28% of the final tour was replanned for the controlled environments and representative tank environments respectively. However, the tour degradation experienced was not significant enough to impact *tour times* or the RCE between the *adaptive coverage planners*. Even when AD-P replanned under the worst-case scenario (92%) the RCE, whilst marginal, was still 1.5 times better than AD-R.

In *Experiment 3*, tour degradation of AD-P solutions did not occur at all, as AD-R was found to produce longer tours due to an increase in covering set sizes. As a result of AD-P achieving significantly faster replanning times and better-quality tours than AD-R, AD-P successfully outperformed AD-R in every planning problem. Therefore, for the tested planning problems, the potential drop in tour optimality that may exist for AD-P did not impact upon the *tour times* compared to AD-R.

Question 3) How does the *plan repair strategy* scale with the size of the environment and size of the detected features?

The controlled environments in *Experiment 1*, showed that regardless to the initial tour used or the size of the environment, AD-P consistency produced equivalent covering set sizes for each identical feature introduced (Figure 8-17). The size of the environment did not impact planning times for AD-P as the replan effort was constrained to work within the defined

ROIs surrounding the new features. The same trends were observed in *Experiment 2* and *3*. As the replanning effort increased, due to the introduction of more features, so did the replan times and the size of the covering sets.

Question 4) What is the replanning limit in which the *full replan strategy* outperforms the *plan repair strategy*?

As AD-P outperformed AD-R in many cases with respect to both planning time and tour quality, these experiments were not able to conclusively find a limit that would render AD-R, in any tested case, to be the better replanning strategy to implement. The *tour times* and RCE values used to benchmark the output of AD-P and AD-R clearly showed the computational advantages AD-P provided over all tested planning problems. As mentioned previously, even when AD-P experienced degradation in tour quality, when the replan effort increased, the degradation experienced did not hinder the overall performance of the *plan repair strategy*, in all tested cases.

While these experiments could not find the limit to which the *plan repair strategy* would be at a significant disadvantage to the *full replan strategy*, the analysis of the these experiments conclude that the *plan repair strategy* is indeed the better of the two replanning strategies to extend the *offline sampling-based coverage planner* to the online domain. This conclusion does not imply that the *plan repair strategy* is the overall best choice for all possible planning problems. There is logical argument to say that if the *plan repair strategy* had to replan 100% of the tour, the additional processes that are required to segment and merge, would render the *full replan strategy* to be the better solution to implement. However, these experiments found that the *segment* and *merge* processes were negligible considering the time these processes saved by limiting the utilisation of the Online CSP and MPP processes, which will increase the more complex the visibility and mobility constraints that are applied. While a definitive limit was not found in these experiments, they did demonstrate that the *plan repair strategy* was capable of replanning up to 92% of the existing tour and still be computationally faster than the *full replan strategy*.

Despite the computational efficiencies of the *plan repair strategy*, the analysis of the intrinsic behaviours of the *adaptive sampling-based coverage planner* revealed the following limitations:

Limitation 1) The *plan repair strategy* must follow the existing order of the preserved tour segments (Section 8.3.2). The deliberate forcing of the new path segments to route between preserved segments may contribute to overall degradation of tour quality.

Limitation 2) The *point-to-line algorithm* (Section 7.6) that is used to sort configurations into sub-MPPs does not consider changes in geometry of the environment. As a result, configurations may be sorted into path segments that are not geometrically appropriate (Section 8.3.2).

Limitation 3) While the *hybrid-heuristic* provided significant improvement of the MPP times, the calculation of the heuristic is computationally inefficient to be used in conjunction with the *plan replan strategy*.

Whilst these limitations existed within the *plan repair strategy*, the degradation of the replan strategy did not at any stage create a solution that was computationally worse than any solution provided by the *full replan strategy* in any tested planning scenario.

In summary, considering the two replanning strategies, the *plan repair strategy* is the superior of the two approaches in this context. The introduction of the ROI Validation phase to prevent non-influenced segments of the tour from being replanned, has made it possible for the sampling and path planning algorithms of the *offline sampling-based coverage planner* to be applied in the online domain. Minimising the amount of work these procedures would be required to perform under a *full replan strategy* has significantly reduced the time it takes to replan new features into the coverage plan.

For the most complex environment tested, *Tank-P4, AD-P was able to complete the entire replanning problem within 7.5 minutes, 379 times faster than AD-R, which solved Tank-P4 in 46 hours.* This result alone, clearly illustrated that the *adaptive sampling-based coverage planner* that utilises a *plan repair strategy* is capable of replanning new coverage in both large and complex environments. Finally, as AD-P's ARTIP were within the seconds, it was suitably under the required one to two-minute replanning time specified by the STIPP criteria. Meeting this final requirement ensured that the *plan repair strategy* was able to satisfy all the STIPP criteria and thereby satisfy the objective of this thesis.

Chapter 9
Summary, Original Contributions, Conclusions, and Future Work

9.1 Thesis Summary

The objective of this thesis was to develop an *adaptive coverage planner* that enables an autonomous robot to perform a thorough visual inspection of the ballast tanks inside a Collins Class submarine. Limitations in the current robotic platform technology make it challenging to solve the autonomous inspection problem inside these complex confined spaces. A review of robotic inspection platforms suggested that a multi-legged, high degree-of-freedom (DOF) robotic platform with a magnetic climbing ability would be a suitable platform for submarine tank inspection. A concept demonstrator was presented that was to be the testbed for algorithm development.

To enable a robotic platform to perform an autonomous inspection, three processes were highlighted:

1) *a mapping system*,
2) *an inspection planner*, and
3) *a motion planner*.

Given that knowledge about the submarine tanks were known *a priori*, a *combined planning architecture* was chosen that allowed an offline inspection plan to be generated first which then initialised an *online planning architecture* capable of modifying the existing plan to account for changing conditions. This approach was considered to be more appropriate than a complete online solution that incrementally constructs a plan in real-time as the robot finds its way through an unknown environment.

The *combined planning architecture* created the Inspection Planning Framework (IPF) that was to be implemented on the concept demonstrator. The three processes were separated into three modules, each responsible for ensuring the robot was capable of executing the calculated inspection plan.

Of the three IPF modules, this thesis focussed on the development of an *adaptive coverage planner*. The *adaptive coverage planner* ensured the *inspection planner module* was able to deliver consistent and reliable planning updates to the robot so newly detected features were included in the inspection. A list of requirements was presented that formed the *submarine tank inspection planning problem* (STIPP), that was the basis of the investigation of this thesis. The STIPP criteria outlined what is required by the coverage planner to ensure it is suitable to be deployed on multi-legged platform. However, it was the delivery of timely planning updates that created the scope and focus of the thesis.

In *Chapter 2*, a review of the path and *coverage path planning* (CPP) literature found that many current coverage path planning solutions did not completely meet all the required STIPP criteria. Given the requirements of the STIPP, it was concluded that a discrete sampling-based approach to coverage planning was best suited for the application. However, no online sampling-based coverage planners were available that met the STIPP criteria. Therefore, a *gap statement* was derived, and it was determined that, to meet all the criteria of the STIPP, an offline sampling-based coverage planner should be converted to the online domain using the common replanning strategies of path planning.

Given the *gap statement*, *Chapter 3* formulated a solution to STIPP. On merit, the *offline sampling-based coverage planner* of Englot and Hover (2017) that uses *redundant roadmaps* to generate coverage was selected to be the discrete coverage planner and was adapted for online implementation. While online implementations exist that use the sampling and path planning procedures for the coverage in unknown environments, there has been no previous attempt to create an *adaptive coverage planner* that modifies an existing plan in a partially known environment. With this coverage planner in mind, two replanning strategies were proposed to adapt the *offline coverage planner* into the online domain: a *full replan* and a *plan repair*. The focus of this thesis was to determine which is better suited to solve the online planning problem.

Chapter 3 also discussed concerns about adapting the *offline coverage planner* to the online domain that were raised in the literature. Emphases on the coverage planner's long

computational times impacting online performance was the primary concern. However, given that there was no available benchmark for how the *offline sampling-based coverage planner* would perform in the target environment, it was decided to perform an experiment to understand its intrinsic behaviour.

Chapter 3 concludes with a discussion of the assumptions placed on the relationship between the three modules of the IPF. These assumptions constrained the investigation of the thesis to only focus upon the development of the *adaptive sampling-based coverage planner* irrespective of any particular mapping system, robotic platform or visibility constraints. These assumptions were:

Environment, Mapping System and Adaptive Replanning

Assumption 1: Newly detected features all exist within the bounds of the original environment.

Assumption 2: Changes within the environment are expected to be small in relation to the size and complexity of the environment.

Assumption 3: The mapping system will only introduce new features into the environment. Given the nature of online mapping, partial maps are also expected.

Assumption 4: No features within the environment are dynamic in nature.

Assumption 5: The *adaptive sampling-based coverage planner* will not handle mesh, sensor and positional uncertainty.

Assumption 6: The *adaptive sampling-based coverage planner* is not responsible for uncovering or directing the robot to uncover new surfaces of partially constructed features.

Assumption 7: Replanning can occur simultaneously to performing the inspection.

Motion Planning and Visibility Constraints

Assumption 8: A 6-DOF holonomic assumption is used as a heuristic to approximate the motion constraints of an 18-DOF robot with electromagnetic adhesion.

Assumption 9: High-fidelity motion planning will occur after the coverage plan has been calculated to reduce the impact on planning times. The *sampling-based coverage planner* will solve motion plans with the simplified constraints of

Assumption 8. Therefore, the tour provided by the coverage planner approximates the actual tour.

Assumption 10: Visibility constraints are simplified to aid the high-fidelity motion planner to achieve the calculated 6-DOF by relaxing the roll constraint of the viewing pose.

In *Chapter 4*, the *offline sampling-based coverage planner* was examined to gain an understanding of any shortcomings needing to be addressed for an efficient online implementation. To test the offline coverage planner, a series of planning environments, including those of representative tanks, were used to evaluate different aspects of algorithms behind the *covering sampling problem* (CSP) and *multi-goal planning problem* (MPP).

The *offline coverage planner* was recreated in this thesis to the specifications outlined in the original publications. To prepare for online planning, amendments were made to increase the delivery of solutions from the CSP and MPP processes. These included;

1) a *primitive rejection limit* to ensure that *unobservable primitives* are not continually sampled after a number of attempts have been made to view them, and
2) a change in approximate *Travelling Salesman Problem* (TSP) solvers that should produce faster TSP solutions for larger planning problems.

The analysis of the benchmark experiment showed that;

1) a significant variability was present between overall planning times for identical covering sets despite producing tours of similar quality, and
2) the coverage planner can be sensitive to small changes within the environment.

The cause behind the variability and sensitivity was narrowed down to the *lazy point-to-point planner* (LPP), confirmed by the long computational times present among the larger planning problems. The sampling algorithms to solve the CSP were not deemed to be of concern for online implementation. Therefore, it was clear from these observations that if an efficient online implementation was to be derived from the *offline sampling-based coverage planner*, the significant variability and sensitivity of the LPP had to be addressed, and either resolved or minimised

Chapter 5 investigated the LPP data from the benchmark experiment and determined that the cause of the variability was due to the *equality termination condition* not compensating

for non-optimal solutions provided by approximation TSP solvers. Given that approximation TSP solvers do not guarantee optimal solutions, the likelihood of terminating due to the *equality termination condition* was more unlikely, as the covering set grew in size. Ultimately, the LPP was found to be terminating at random.

To minimise the variability, whilst still retaining an approximation TSP solver to solve TSP problems quickly, *additional termination conditions* were developed to track the two primary behaviours observed in the planning data;

1) the non-consecutive repetition of previous solutions in small planning problems, and
2) the delivery of multiple yet not repeating sub-optimal answers presented in large planning problems.

The implementation of the *additional termination conditions* minimised the variability exhibited by the LPP resulting in shorter planning times with no degradation to tour length. The *larger planning problems* that had failed to find a solution within six hours were completed within 10 minutes.

By removing the variability from the planning solutions, further examination of the LPP data showed how sensitive the LPP is to the environment. Results showed that there were a significant number of RRTs being evaluated that were not retained in the final plan. This highlighted the significant effort the LPP is required to make to find a feasible solution. Considering the expense of solving high-fidelity motion plans, a reduction in the number of path evaluations would be beneficial for both offline and online planning approaches.

Chapter 6 explains how the sensitivity of the LPP was due to the underestimation of the *Euclidean assumption* that is used to initialise the connectivity between the configurations before planning. To overcome the naivety of the *Euclidean assumption*, it was concluded that the adjacency between the configurations had to encode a representation of the environment.

Topological skeletons were used to create the *skeleton-heuristic* that provided an approximation of the distances between configurations around obstacles within the environment. Rerunning the benchmark experiment showed that the while the *skeleton-heuristic* reduced planning times and the number of paths evaluated, its distance metric degraded tour quality.

Due to the *thinning process*, configurations that are could have been connected via line-of-sight had their distances overestimated therefore making it seem that they were further away than they appeared. To overcome the overestimation within local areas of connectivity, the *Euclidean assumption* was reinstated between line-of-sight configurations and thus created the *hybrid-heuristic*. The *hybrid-heuristic* was found to significantly decrease planning times in the large planning problems to within minutes by removing thousands of paths from being evaluated without significantly degrading tour quality.

To demonstrate the effectiveness of the *hybrid-heuristic* over large planning environments, it was then applied to a life-like 70x40x2m office space. The performance of the *hybrid-heuristic* excelled in comparison to the LPP initialised with a *Euclidean assumption*. Feasible tours with minimal degradation were generated in under 10 minutes compared to 1.1 days using the *Euclidean assumption*. This result demonstrated that the *offline sampling-based coverage planner* is capable of solving significantly larger planning problems with the *hybrid-heuristic*, thus overcoming one of the major concerns with the *offline sampling-based coverage planner*.

At the conclusion of *Chapter 6*, two new *heuristics* where introduced to the LPP to reduce *multi-goal planning problem* (MPP) times. Despite the significant improvement provided by the *additional termination conditions* and the *hybrid-heuristic*, it was not sufficient to determine that the *full replan strategy* would be a suitable extension for online planning. The *plan repair strategy* would prove to be a better strategy providing the segmentation process did not significantly impact the planning times.

Chapter 7 proposed the *plan repair strategy* to develop the *adaptive sampling-based coverage planner* that would meet the STIPP criteria. To minimise the overall replanning effort, the *plan repair strategy* formed *regions of interest* (ROIs) around newly detected primitives to isolate the replanning effort. Treating each primitive as a separate ROI allowed multiple regions to be replanned concurrently, irrespective of their geometric position. Setting the ROI threshold to the inspection sensor's maximum field-of-depth ensured all *potentially unobserved primitives* were included in the replanning phase without having to parameterise the environment to perform effectively.

Introducing ROIs to isolate replanning regions by segmenting the current plan created a three-phase coverage planner that still preserved the existing offline CSP and MPP procedures. The proposed *adaptive sampling-based coverage planner* introduced the

ROI Validation phase, to supply the *Online CSP* and *Online MPP* phases with smaller planning problems. Given the *plan repair replan strategy* preserved the functionality of the CSP and MPP processes, the *adaptive coverage planner* still adhered to probabilistic completeness. Hence, each ROI captured all potential primitives that may be under the influence of the changes.

Chapter 7 also introduced the *optimal sampling procedure* (OSP) to aid in producing consistent viewing locations before resulting random samples to fill coverage over simple geometries. Furthermore, this chapter introduced the *relaxed-trapped configuration heuristic* (R-TCH) and *trapped configuration heuristic for path planning* (TCH-PP) to remove configurations that are trapped due to *prison cells* created by an evolving representation of the environment.

Chapter 8 compared the two *replanning strategies* to determine if the *plan repair strategy* (AD-P) is the better of the two replanning strategies for extending the *offline sampling-based coverage planner* to the online domain. For completeness, the *adaptive coverage planner* using a *full replan strategy* (AD-R) was also proposed. The chapter aimed to answer the following questions that would determine the suitability of AD-P to adaptively solve replanning queries:

Question 1) *What is the computational impact of segmentation and merging of sub-plans on the planning times*?

Question 2) *What is the potential drop in tour optimality as the sub-plan approach does not solve the coverage plan globally?*

Question 3) *How does the adaptive coverage planner, using the plan repair strategy, scale to the size of the environment and size of any changes?*

Question 4) *What is the replanning limit in which the full replan strategy outperforms the plan repair strategy?*

To answer these questions a series of simulated experiments were conducted that analysed the behaviour of both AD-P and AD-R within simplified and representative tank environments to both static and evolving environments. The primary experiment documented in *Chapter 8* examined the response of *adaptive coverage planners* to an environment that scaled in the size and number of objects. The main difference between the two adaptive coverage planners was the amount of replanning that was required to perform an update. The question with the *plan repair strategy* was whether additional computation

cost from the *segment* (ROI Validation) and *merge* (Online MPP) processes would outweigh the potential benefits of replanning smaller planning problems faster.

The outcome of the size and scalability experiment determined that for the simplest replanning problems, the *adaptive coverage planner*, using a *plan repair strategy* (AD-P), was the superior of the two approaches. The introduction of a segmentation process significantly reduced the replanning effort required, resulting in shorter replanning times. Neither the segmentation nor merging processes of the ROI Validation and Online MPP phases negatively impacted planning times.

Subsequent experiments in the representative tank environment demonstrated the same trends in behaviours between the two *adaptive coverage planners*. The *plan repair strategy* was better suited to performing replanning within the target environment. When simulating the robot executing the inspection plan, the differences between the two replanning strategies grew. AD-R was found not to be suitable for solving the online planning problem. When planning over *Tank-P4*, AD-P solved the replanning problem 379 times faster than AD-R, reducing replanning times from 46 hours to 7.5 minutes. With this result, the *plan repair strategy* was deemed to be the superior of the two replanning strategies to convert the *offline sampling-based coverage planner* to the online domain.

These experiments also tested the effectiveness of the *additional termination conditions* and *hybrid-heuristic* in the online context. While it was evident that the *additional termination conditions* functioned as expected, the calculation of the skeleton outweighed the benefits it provided in solving the MPP faster than using the *Euclidean assumption*.

9.2 Original Contributions

This thesis makes the following five original contributions to the field of coverage path planning:

Contribution 1: The thesis investigated the functionality of the *offline sampling-based coverage planner* developed by Englot and Hover (2017). The investigation determined the suitability of sampling and path planning processes for an online implementation (*Chapter 4*). The analysis of experimental data revealed that major issues were present when using large covering set sizes numbering into the thousands. These issues made it infeasible for problems of this size to be solved appropriately and thus necessitated new directions of development to improve both

the offline and online planning processes (*Contributions 2 and 3*) The original contribution to knowledge is the analysis and findings of this experiment. This experiment examined the *offline sampling-based coverage planner* in a different planning domain with different planning constraints, providing further insight about the coverage planner than previously published.

Contribution 2: The investigation in *Chapter 4* revealed that the existing *offline sampling-based coverage planner* exhibited significant variability between solutions when planning over large covering sets. This thesis explored the cause behind the variability and found that it was produced by the *equality termination condition* and approximation *Travelling Salesman Problem* solutions within the *lazy point-to-point planner* (*Chapter 5*). To rectify this variability, *additional termination conditions* were developed. These new termination conditions successfully removed the majority of the planning time variability due to ineffective iterations, which consequently resulted in a significant reduction of planning times across all planning problems. These *additional termination conditions* enabled both the *offline* and *adaptive sampling-based coverage planner* to continue using an approximation TSP solver to solve large coverage planning problems quickly.

Contribution 3: The analysis in *Chapter 4* also demonstrated the sensitivity of the *lazy point-to-point planner* to the geometry of different environments. Topological skeletons were introduced to create the *hybrid-heuristic* and increase the efficiency of the *lazy point-to-point planner* to better solve the *multi-goal planning problem* in complex spaces (*Chapter 6*). The *hybrid-heuristic* generally guided the *lazy point-to-point planner* to a faster convergence on solutions, significantly reducing planning times of large complex coverage planning problems in concave planning environments.

Contribution 4: This thesis presented a novel *adaptive sampling-based coverage planner* that is capable of performing inspections from an autonomous platform within confined, complex environments (*Chapter 7*). An adaptive variant of the *offline sampling-based coverage planner* of Englot and Hover (2017) was built by extending the planner with the capability to modify the current inspection plan to accommodate new changes within the environment (*Chapter 3*). The thesis investigated two replanning strategies, a *full replan* and a *plan repair* to extend the

offline coverage planner into the online domain. The investigation found that the *adaptive coverage planner*, using a *plan repair strategy*, significantly reduced the computational effort required to update an existing inspection plan compared to an *adaptive coverage planner* using a *full replan strategy* (*Chapter 8*).

Contribution 5: While a *plan repair strategy* using a *region of interest* to bound the influence of change is not a novel approach to minimising the replanning effort, the application of a *plan repair strategy* to the *offline sampling-based coverage planner* has not, to the author's knowledge, been attempted previously (*Chapters 3 and 7*). Application of the *region of interest* based on the sensing capability of the visual sensor enabled the planning processes of a state-of-the-art *sampling-based coverage planner* (which has been highlighted as an expensive coverage planner and inappropriate for direct online implementation) to perform efficiently online.

9.3 Conclusions

The objective of this thesis was to develop an *adaptive coverage planner* that met a list of criteria to enable a complete and comprehensive inspection of the submarine tanks on-board the Australian Collins Class submarines by a multi-legged, high-DOF robot. This thesis addresses the STIPP criteria that were defined to ensure the *adaptive coverage planner* was suitable to be implemented as the *inspection planning module* within the IPF. The following section describes how the ideas developed in this thesis satisfy the STIPP criteria and concludes on the major findings that indicate how these requirements were met.

Requirement 1: Construct an inspection plan from a known 3D model of the environment.

For the situation with no damage or modifications to the structure of the tanks, the CAD data of the tanks was considered sufficient to generate an offline plan. After a review of the literature, the *offline sampling-based coverage planner* of Englot and Hover (2017) was chosen as the most applicable *coverage planner* to meet the STIPP criteria given that there was no adaptive discrete coverage planner available that met all the requirements.

Requirement 2: The inspection plan should consist of discrete viewing positions that enable the robot to stop and photograph desired surfaces.

Both the *offline* and *online coverage planner* are derived from sampling-based coverage techniques that create a series of discrete viewing locations, enabling the robot to stabilise

at each vertex of the plan and acquire coverage. Given the complexities of motion planning for a multi-legged, high-DOF platform, it was concluded that acquiring coverage at discrete locations was more desirable than using continuous CPP techniques that acquire coverage over the edges of the plan.

Requirement 3: The *inspection planning module* should implement a coverage planner that has the ability to generate an inspection plan for a variety of different tank variations without parameterisation of the environment.

The selection of the *offline sampling-based coverage planner* that uses *redundant roadmaps* was a suitable candidate due to the generic nature of the algorithm. The *sampling-based approach* to coverage planning requires no parameterisation or decomposition of the environment to formulate a solution. As the coverage was determined directly from the visibility and mobility constraints of the robot, the coverage planner could calculate the robot's achievable coverage for any environment. The *sampling-based coverage planner* was therefore a suitable candidate to solve the coverage plans in submarine tanks where explicit parameterisation was forbidden.

Requirement 4: The *inspection planning module* should assure either complete coverage or the highest attainable coverage of the interior tank surfaces, including all internal fittings and reinforcement structures.

Considering the complexity of the intended planning environments and limitations in the robotic platform, there is a significant probability that it may not be possible to physically observe all primitives. To compensate for the potential loss of coverage, the 100% coverage constraint was relaxed, providing *unobservable primitives* were reported appropriately.

Additional procedures were developed to record the status of all detected primitives, especially the ones which could not be sufficiently observed by the camera. These developments took the form of the *primitive rejection limit*, to limit sampling on *unobservable primitives* and the *unobservable primitive list*, by recording all primitives in the planning problem that could not be observed (Section 4.2.3). Considering the consistency of the CSP times in the realistic office space experiment (Section 6.9), where some primitives were classified as *unobservable*, it was sufficient to conclude that these trivial improvements ensured all primitives of the environment were taken into account without significantly impacting the CSP procedures.

Requirement 5: The *inspection planning module* should generate coverage plans that accommodate a multi-legged, high-DOF robotic platform.

To accommodate the development of inspection plans for a multi-legged platform, this thesis accomplished the following:

1) Selected an appropriate coverage planner that could generate inspection plans with minimal issues for a platform with complex mobility constraints. The decoupled planning approach to coverage planning was sufficient for a multi-legged robot that possesses, or at least approximates, holonomic capability.

2) Simplified mobility and visibility constraints were implemented to alleviate the computational burden of generating the platform's motion plans within the *adaptive coverage planner*. Within the IPF, high-fidelity motion planning only occurs over the final edges of the inspection plan instead of on each edge evaluated by the LPP. From an analysis perspective, the reduction of these constraints allowed for a deeper understanding of the planning procedures without the influence of a particular robotic platform, while but still accommodating the intended platform.

3) To reduce the LPP's computation, additional algorithms were required to generate a better initial representation of the connectivity between configurations. The *hybrid-heuristic* was developed to better inform the LPP of the connectivity between configurations, minimising the requirement of a motion planner when planning in a lazy context. Reducing the number of motion planning queries reduced the computational effort of the LPP. This finding is particularly useful if high-fidelity motion planning was to be undertaken within the LPP and this should be generalisable to other applications where the LPP is utilised.

Requirement 6: The *inspection planning module* should contain an internal framework that allows adapting an existing offline inspection plan for coverage of newly detected features.

As discussed in *Chapter 1*, it was decided to implement a *combined planning architecture* on the platform given that known data about the submarine tanks was available. As the robot would not use traditional online exploration algorithms to perform the inspection, an adaptive approach to coverage planning was taken. It was concluded that a sampling-based approach was best suited to meet the problem criteria, given the current literature did not

meet all STIPP requirements.

Generating an inspection plan was handled over two processes. As discussed in Section 3.4.2, decoupled planning approaches typically make planning for non-holonomic robotic platforms difficult. Since, the target platform is holonomic in nature, it was decided to keep the two-phase planning approach. This ensured that each of the two processes could be individually modified for online use without impacting the other.

The *adaptive sampling-based coverage planner* using a *plan repair strategy* was constructed on the principle that the two phases were appropriate to solve the online planning problem provided a suitable segmentation process could minimise the replan effort. The novelty of this approach was introducing the ROI Validation phase which segmented the current plan based on the sensor's visibility constraints. The additional phase fit well with the Online CSP and Online MPP phases to maintain an isolated but coherent replanning process.

The Online CSP and Online MPP were developed to interface with the existing CSP and MPP processes without modifying them. Online CSP still creates redundant roadmaps while Online solves the MPP. The difference to the traditional offline procedures is that Online CSP and Online MPP solve over smaller sub-sets of the overall environment and inspection plans that has been defined by ROIs created in the ROI Validation phase. Collectively the three phases the plan repair strategy were able to efficiently replan complex 3D environments.

Requirement 7: The *inspection planning module* should provide inspection plan updates in a timely manner.

The primary investigation of the thesis was to determine which of the *full replan* or *plan repair* strategies was best suited to extend the *offline sampling-based coverage planner* into the online domain. Consequently, the majority of the thesis focussed on fulfilling the requirement to provide timely planning updates. This was achieved by investigating the intrinsic properties of the *offline sampling-based coverage planner* and then adapting its processes to work in an online situation. It was decided that, as there was no prior research available on how a multi-legged platform would perform an inspection in the Collins Class submarine, replanning times in the *minutes* would be acceptable, and would be particularly desirable if kept to under two minutes (Section 1.5.1).

It was important to undertake the benchmark experiment because there was no current

research that indicated how the *offline sampling-based coverage planner* would perform in the target environment. Despite the coverage planner's known limitations, the benchmark experiment confirmed that the recreated *offline sampling-based coverage planner* was not suitable for direct online implementation. It was concluded that the variable planning times of up to six hours in the representative tank environments and the sensitivity of the LPP to different environments needed to be addressed and resolved. Neither replanning strategy would have been successful under these conditions. To minimise the influence of these factors from the planning processes the *additional termination conditions* and *hybrid-heuristic* were developed.

When investigating the use of topological skeletons resolve the sensitivity the LPP has to different environments, it was found that using skeleton distances alone were not suitable at representing the connectivity of locally connect free-space branches. By reinstating the *Euclidean assumption* to appropriately represent the connectivity between line-of-sight configurations enabled the *hybrid-heuristic* to significantly reduce the number of path evaluations required to be solved in complex environments.

The analysis of the *additional termination conditions* and *hybrid-heuristic* demonstrated their effectiveness at improving offline planning times in large and complex environments. Despite significant improvements to overall planning times, that saw some planning problems reduced from six hours to seven minutes, it was not enough to safely ensure a *full replan strategy* would be an effective approach to implement online. Offline planning times were not within the 1-2-minute timeframe that would warrant time for the *full replan strategy* to be feasible over a long inspection that could contain several changes. Therefore, it was concluded from these investigations that the *plan repair strategy* may prove to be the better replanning strategy to apply.

For online planning, it was concluded that the *additional termination conditions* were essential for reliable planning updates in the online context. However, while topological skeletons aided path planning within environments where the *Euclidean assumption* was not appropriate, the calculation of the skeleton was an expensive process. This was demonstrated in *Experiment 2* (Section 8.4) where the cost of calculating *hybrid-heuristic* for AD-P(HY) outweighed its benefits for all tested environments.

The experimental results in *Chapter 8* demonstrated that the *plan repair strategy* was the most suitable approach of the two replanning strategies to extend the *offline sampling-based*

coverage planner into the online domain. The introduction of the ROI Validation phase was sufficient to segment the planning problem into smaller sizes. Furthermore, as the *plan repair strategy* generated smaller planning problems, the new Online CSP and Online MPP procedures solved the replanning problem faster than the *full replan strategy* on all tested scenarios.

In the size and scalability experiment (*Experiment 1*), the *plan repair strategy* could replan up to 92% of the current tour and still produced solutions in a less time than the *full replan strategy*. The ability to replan upwards of 92% of the existing tour exceeded initial expectations that sought to develop an *adaptive coverage planner* that could handle at least 30% of change of the environment (*Assumption 2*). The trends observed in *Experiment 1* continued to be observed across the other two experiments as the gap in performance grew between the two replanning strategies as the complexity of environment increased.

While a definitive limit to when the *plan repair strategy* degrades in performance to the *full replan strategy* was not found, it can be certain that the *full replan strategy* would outperform the *plan repair strategy* when the entire tour requires replanning. The replan effort is the same but additional overheads to segment and merge increase the total replan time. Even if the mobility and visibility constraints of the platform became more complex, the difference between the *full replan strategy* and the *plan repair strategy* would only increase further, as the *full replan strategy* will always do more planning than the *plan repair strategy*.

The final experiment showed the advantages of the *plan repair strategy* as it was able to replan the entirety of *Tank-P4* in the time it took for the *full replan strategy* to complete a single replanning iteration. AD-P's overall replanning time of 7.3 minutes with an average replan time per map update of 1.52 seconds against AD-R's 46 hours and 7.5 minute replan time per iteration was enough to conclude that the *plan repair strategy* was the better strategy to convert the *offline sampling-based coverage planner* to the online domain.

Finally, it was concluded from this research that the following limitations were present in the ideas and work developed in this thesis:

Limitation 1) The *plan repair strategy* must follow the existing order of the preserved tour segments (Section 8.3.2). The deliberate forcing of the new path segments to route between preserved segments may contribute to overall degradation of tour quality.

Limitation 2) The *point-to-line algorithm* (Section 7.6) that is used to sort configurations into sub-MPPs does not consider changes in geometry of the environment. As a result, configurations may be sorted into path segments that are not geometrically appropriate (Section 8.3.2).

Limitation 3) While the *hybrid-heuristic* provided significant improvement of MPP times, the heuristic is computationally inefficient, due to computing the skeleton, to be used in conjunction with the *plan replan strategy*.

Limitation 4) The *adaptive sampling-based coverage planner* is not applicable to autonomous robotic platforms that do not possess holonomic ability or stability control (Section 3.6.2). Given the *adaptive coverage planner* preserves the decoupled sampling and path planning processes, the coverage planner has the same limitation as its offline counterpart.

Limitation 5) As the *online motion planner* was not available at the time of the experimental trials, the scheduled physical trials did not occur. The *adaptive sampling-based coverage planner* was not tested with robotic platform constraints of the concept demonstrator. Therefore, only simulated analysis was performed (*Chapter 8*).

Limitation 6) The results of this thesis do not completely encapsulate the constraints of a multi-legged, high-DOF robot. A generalised 6-DOF holonomic representation was used as the mobility model in all experiments (Section 3.6). Therefore, the results of the thesis are more generic and applicable to all robots of this capability at the expense of a plan customised to the particulars of the target platform.

Despite the listed limitations, this thesis has achieved its aim by developing a *3D adaptive sampling-based coverage planner* that in simulation at least, is capable of producing stable and consistent planning updates within confined complex environments. This thesis has also provided additional improvements in the form of the *additional termination conditions* and the *hybrid-heuristic* that enable both the *offline sampling-based coverage planner* and *adaptive sampling-based coverage planner* to solve large complex planning problems.

9.4 Future Work

This thesis explored several avenues to create an *adaptive sampling-based coverage planner* suitable enough to perform inspections inside the complex spaces inside the Australian

Collins Class submarine tanks. However, as discussed in Sections 9.1 and 9.3, there where are several assumptions that were placed on the development of the *adaptive coverage planner* and discovered limitations of the presented solution that leads to many compelling areas of future work. The future work presented in this section reconsiders certain aspects of the proposed methodologies presented in this thesis, as well as providing extensions to these ideas that may yield better outcomes. The following sections highlight areas of future work for the;

1) *additional termination conditions,*
2) *hybrid-heuristic, and*
3) *adaptive sampling-based coverage planner using a plan repair strategy.*

For each of the ideas listed above the future work is classified as either one or more of the following categories:

1) computational improvement,
2) algorithmic improvement, or
3) quality improvement.

In addition to these improvements, algorithmic development should focus on the improvement of the mapping, coverage planning and motion planner to include positional and sensor uncertainty (*Assumption 5*). If these processes can sufficiently model uncertainty, it will lead to more robust inspections from autonomous platforms. However, techniques to implement this were not investigated and therefore were not discussed in detail in future work.

9.4.1 Additional Termination Conditions: Terminating upon the First Sign of No Improvement

Removing the State of Minimal Improvement: Algorithmic / Computational Improvement

While the *additional termination conditions* produced significant improvements to LPP times across the planning problems, some of the larger planning problems were still subject to minor variability in planning times due to extra iterations required to determine if the LPP is in a *state of minimal improvement* (Table 5-10). To provide the LPP the best opportunity to find a solution, the *additional termination conditions* were designed to accommodate all planning problem sizes. This enabled the LPP to find the best solution without terminating prematurely. However, this also allowed additional iterations to execute before the LPP

terminated. For larger planning problems, up to 50% of the total number of planning iterations reported ZEIs, however compared to how the LPP originally functioned, the impact was minimal.

On reflection, terminating upon the first sign of no significant improvement will remove the state *of minimal improvement* completely. The analysis of the results in *Chapter 5* (Table 5-3) showed that for the planning environments used, termination upon the *First-Zero Path Evaluation Iteration* (FZEI) would generally produce a result within 1% of the best tour length that could be found. Given the sub-optimality present in the planning processes (RRT paths and random sampling etc.), if time is critical and solving the TSP problem for a given planning problem is computationally expensive, then terminating upon the FZEI would still yield an acceptable tour, despite the risk of premature termination.

9.4.2 Improving the ability of the Hybrid-heuristic to Solve the Multi-Goal Planning Problem Faster

Method 1: Parallelisation: Computational Improvement

A downside to using the topological skeletons was the time taken to compute the skeleton (*Limitation 3*). The recalculation of the skeleton at every replanning iteration removed the possibility of it being useful for online implementation despite the significant advantages it provided offline. The time taken to generate a skeleton can be improved by replacing the single CPU implementation of Lee's thinning algorithm with a parallel implementation. The deterministic nature of the thinning process lends itself well to parallelisation (Liu et al., 2014; Bakken and Eliassen, 2017; Zhu et al., 2015). These listed implementations used multi-CPU or GPUs to achieve a computational improvement over a single CPU implementation.

A *parallel implementation* would significantly reduce the overhead time to calculate a skeleton while maintaining the benefits of solving the MPP with the *hybrid-heuristic*. A parallel implementation would make it possible to produce a skeleton in larger 3D environments and with the potential for the *hybrid-heuristic* to be effective for online implementation.

Method 2: Mesh-based Skeleton Algorithms: Algorithmic Improvement

Another suggestion to reduce the overhead time to create a skeleton is to remove both the voxelisation and thinning processes by using a mesh-based skeleton approach (Cornea et al, 2007; Jalba, Kustra and Telea, 2012). A mesh-based skeleton may yield a better

computational outcome as the representation of the environment is already in the form of a mesh and it will not have to be continually converted to voxels and thinned.

Method 3: Connecting Configurations to Skeletons via Watersheds: Algorithmic Improvement

This thesis proposed an additional rule to the binary thinning algorithm of Lee et al. (1994) to ensure that the voxels containing a configuration were not removed and subsequently added to the skeleton. However, applying this rule may not be applicable to other types of thinning methods. In Section 6.5.1, the use of *watersheds* was proposed as an alternative approach to connect configurations to a skeleton.

While the application of a watershed introduces an additional step to connect the configurations to the skeleton, the approach should allow any thinning algorithm the ability to utilise the *hybrid-heuristic*. Figure 9-1 illustrates what a watershed would look like over the 3D office space used in Section 6.9. Each cluster of the watershed can be used to connect the configurations to the skeleton, via point-to-line evaluations.

Method 4: Converting Large Coverage Planning Problem into a Covering Travelling Salesman Problem: Algorithmic / Computational Improvement

Expanding upon the previous suggestion to use a watershed to connect configurations to a skeleton, it should be possible to use the *clusters* of the watershed to create a *Covering Travelling Salesman Problem* (Current and Schilling, 1989) to solve the coverage planning problem. Figure 9-2 illustrates this potential solution. Each *cluster* creates a smaller MPPs that can be solved individually, as done with the *plan repair strategy*. A final inspection plan can be constructed by solving the MPP between *clusters*.

As the MPP is not considered globally, this proposed solution should evaluate fewer paths, further aiding the coverage planner to solve large complex coverage planning problems. However, as the tour is not considered globally, tour degradation is expected.

Method 5: Better Distance Estimations for Robots Constrained to Moving Over the Inspecting Surfaces: Algorithmic / Computational / Quality Improvement

The experiments in *Chapter 6*, demonstrated the ability of the *hybrid-heuristic* to solve coverage plans for a robot possessing a 6-DOF holonomic capability that can move freely through free space. As such, the skeletons that were generated through the empty regions, as seen with the representative tank model (Figure 6-10), were a suitable approximation of the connectivity between configurations. However, these paths through these regions are not

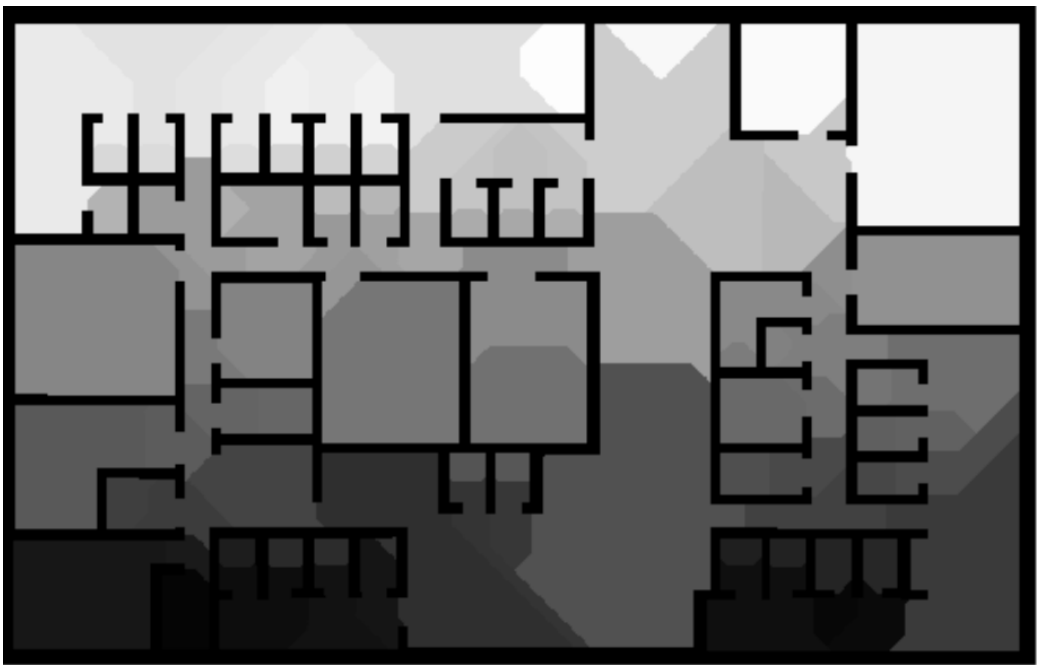


Figure 9-1: Using watersheds to connect configurations to a skeleton. Each shaded region would cluster together groups of configurations to form *a Covering Travelling Salesman Problem*. Watershed over office floor space courtesy of Jonathan Wheare.

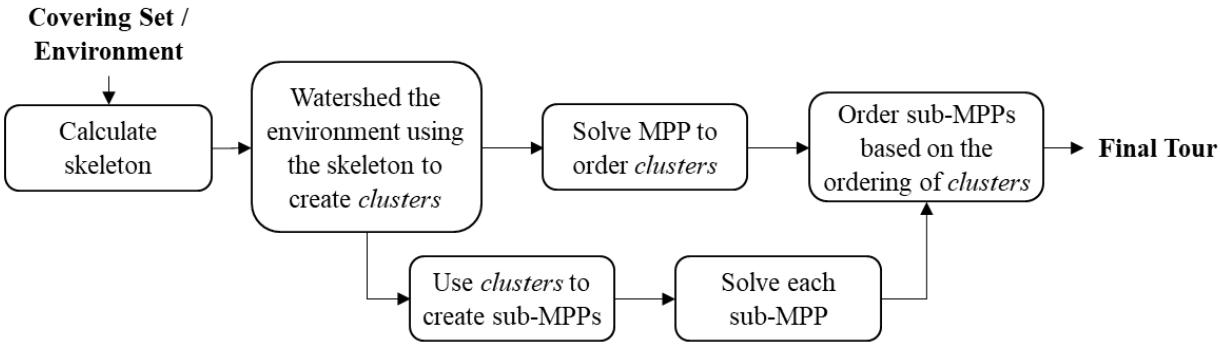


Figure 9-2: Proposed solution to use topological skeletons and watersheds to create a *Covering Travelling Salesman Problem* (TSP) that can be used to solve smaller *multi-goal planning problems* (MPPs) in large complex planning environments.

representative of the paths that a multi-legged robot that is constrained to move along surface would take (*Limitation 6*).

A better approximation would be obtained by removing the non-traversable space from the thinning process (Figure 9-3). This can be achieved by thinning between the actual surface and an inflated surface that encapsulates the mobility constraints of the robot. To include all the configurations into the skeleton, the surface should be inflated to at least the maximum sensor distance (FOD_{max}). Thinning between the actual surface and inflated surface may yield a better approximation and faster thinning times as the open space voxels that cannot be traversed will be removed from the thinning process.

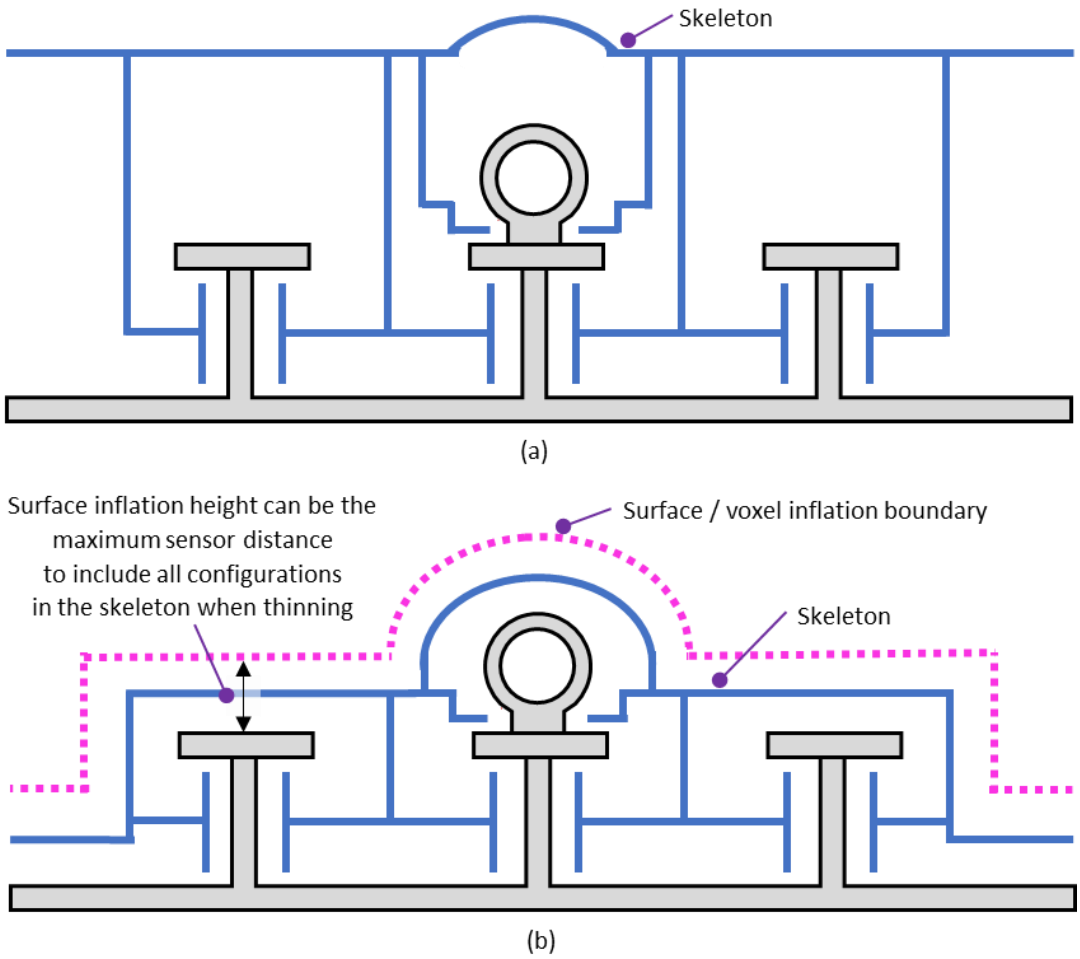


Figure 9-3: The difference in skeletons produced in free space (a) and a skeleton bound to be constructed within an inflated surface boundary (b).

9.4.3 Expanding the Capability of the Plan Replan Strategy

Method 1: Adaptive Coverage Planning in Non-concave Environments: Algorithmic Improvement

Assumptions 1 and *6* require that all the replanning occurs within the bounds of the original environment. However, there is a possibility to extend the capability of the *adaptive coverage planner* to work within non-concave environments.

In the IPF, the *adaptive coverage planner* only replans over new map updates supplied by the *mapping system*. The *mapping system module* is required to inform the *executive system* to direct the robot towards undiscovered areas using exploration algorithms e.g. Next-Best-View algorithms. For non-concave environments, discovering new areas using such algorithms is pivotal to ensure a complete, high-quality inspection can be undertaken in non-concave environments.

Algorithmically, for the *adaptive coverage planner* to accommodate for non-concave

environments, the R-TCH and NNH (Section 7.4.2) will have to be reconsidered. These heuristics work on the assumption that there is an equal valued mesh manifold in each of the cardinal directions. Partially constructed features can force these heuristics to fail and depend upon the TCH-PP to remove these configurations during the path planning phase. However, relying upon the TCH-PP to remove configurations should only be used as a fail-safe approach. If mobility constraints increase in complexity, removing configurations using TCH-PP will become more expensive. An alternative solution is required to ensure that *trapped configurations* are appropriately removed during the sampling phase.

Method 2: Replanning Moved, Modified and Removed Environmental Features: Algorithmic Improvement

Assumption 3 assumed that only new primitives were added into the planning problem. Under this assumption primitives that have moved from their existing position, modified or removed primitives were not included. The natural extension for both the *mapping system* and the *adaptive coverage planner* would be to accommodate for planning problems that possess primitives pertaining to structures that have either moved, been removed or have been slightly modified from the existing representation of the environment.

To implement this extension, primitive indexing must be consistent between the *adaptive coverage planner* and the *mapping system*. Each configuration contains a list of primitives and if primitives have moved from field of view or have been removed, these primitive indices must be updated. However, consistent index changes may prove to be expensive when replanning over environments that observe significant changes.

To minimise recomputing a new mesh upon each replanning iteration, it would be easier to maintain primitive indexing by never removing primitives of the map even if they no longer exist. Newly detected primitives can be appended to the end of the mapping update as currently performed, but if the *mapping system* detects primitives of the original environment that no longer exist, to maintain consistent indexing for the inspection, the *mapping system* should mark these primitives as *unobservable*. Those primitives marked as *unobservable* will be placed on the *unobservable primitive list* and will remain *unobservable* for the entirety of the inspection. Therefore, post inspection, as the *mapping system* contains the list of primitives that no longer exist, identifying the primitives that are actually *unobservable* can be easily determined since *the unobservable primitive list* will contain the same primitive indices. Providing the *mapping system* maintains consistency between the new and original

primitives, a reconstruction of the mesh, post inspection, should yield consistent indexing.

An alternative to handling modified primitives, within the coverage planner, could be to distort the tour based on the movement of the environment instead of replan planning new configurations to accommodate the changes. Instead of storing configurations as positions in configuration space, the configuration could be stored relative to the random primitive that was selected for that configuration to be generated. Therefore, if the primitives move slightly the position of the configuration within the tour will shift with the movement of the primitives. Consideration will have to be given to ensuring that the robot can still achieve the desired position.

Method 3: Improving the Path Merging Process Using the Hybrid-Heuristic: Algorithmic / Quality Improvement

A discovered limitation of the *plan replan strategy* was the sorting process that is required to place new and existing configurations within ROIs into new path segments (*Limitation 2*). Sorting the new configurations into unresolved path segments is determined by a point-to-line evaluation between successive ROI gate parings (Q^{Entry} and Q^{Exit}). While this evaluation is computationally cheap, it does not consider the impact the changes have on the connectivity between configurations. The current metric selects the ideal path segment based on the *distance* between the configuration and each gate pairing. Consequently, it has no encoded representation of the geometry of the space in the evaluation. Essentially, the assumption is akin to the *Euclidean assumption* that has been used as an initial guess of the LPP. The use of the *hybrid-heuristic*, despite its current computational limitations for online use, could rectify this problem.

For AD-P(HY), the *hybrid-heuristic* was performed just before the MPP was solved (Figure 8-9). The distances were calculated once over the complete covering set (S^*) and the ROI gates and each sub-MPP received the appropriate adjacency matrix from *hybrid-heuristic* to initialise the LPP. A proposed solution is to calculate the *hybrid-heuristic* first and then use the approximated distances to sort the configurations as a replacement to using the *point-to-line evaluation* (Figure 9-4). Given that the *hybrid-heuristic* has been solved prior, each sub-MPP can still utilise the distances from the *hybrid-heuristic* to initialise the MPP. If the skeleton could be computed more quickly, using one of the aforementioned ideas, the *hybrid-heuristic* could lead to an increase in tour quality.

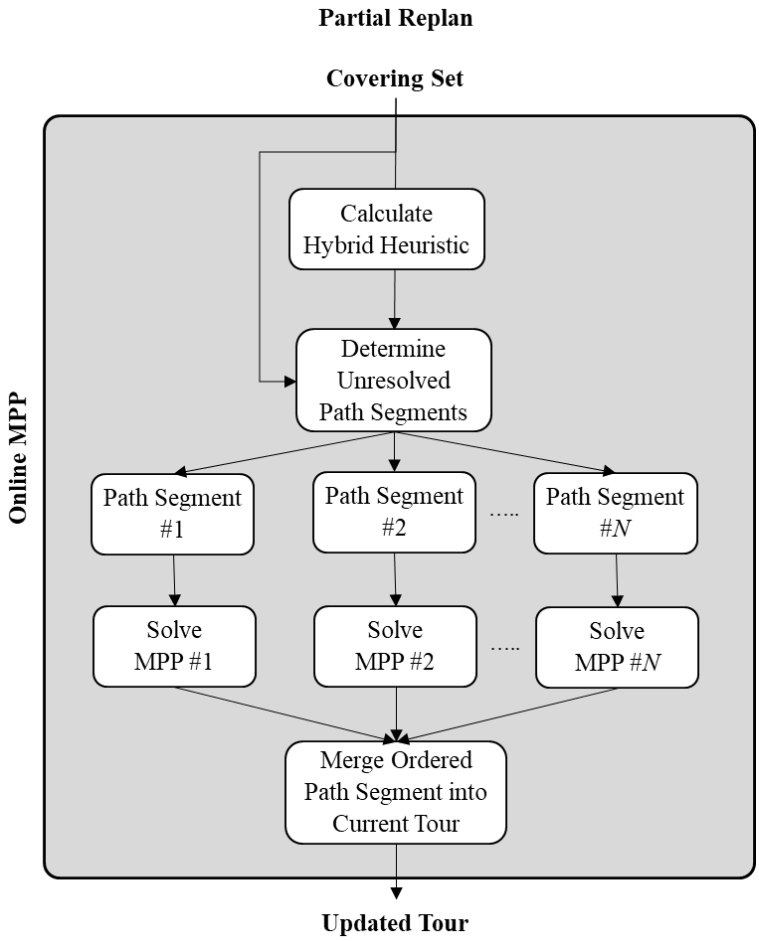


Figure 9-4: Solving the *hybrid-heuristic* first will inform the Online MPP of the adjacency between configurations before sorting the covering set.

Method 4: Parallelisation: Algorithmic / Computational Improvement

To improve the computational efficiency in creating an online tour update, the three phases that comprise the *adaptive sampling-based coverage planner* are independent processes that are inherently parallel in nature. The processes that could benefit from a parallel implementation are list below:

1) Validating each configuration of the tour (ROI Validation).
2) Sampling the new *redundant roadmap* (Online CSP)
3) Solving each sub-MPP using the LPP (Online MPP).

Furthermore, parallelisation of internal algorithms such as;

1) ray tracing to check for primitive occlusion, which is common practice for gaming applications (Schmittler et al., 2005), and
2) generating motion plans for each edge of a TSP solution, are also strong candidates for parallelisation that would significantly improve planning times.

Significant gains could be achieved by concurrently processing the motion plans for each edge of the TSP solver, especially for planning problems that are required high-fidelity solutions. The implementation of parallel processing opens up the opportunity to solve even larger planning problems than those presented in this thesis.

Method 5: Anytime Implementation of the Plan Repair Strategy: Algorithmic / Computational Improvement

The original recommendation, as proposed by Englot (2012), was to extend the *offline sampling-based coverage algorithm* to perform online by implementing the sampling and planning processes as *anytime algorithms*. In this thesis the '*plan as much as time permits philosophy'* was not taken. It was decided that creating a framework for an *adaptive coverage planner* that could solve the complete tour took priority over developing an *adaptive coverage planner* that only relied upon an *anytime implementation* to produce faster planning updates. However, given that the *adaptive sampling-based coverage planner* still retains the functionality of the original offline CSP, SCP and MPP methods, an *anytime implementation* of the *plan repair strategy* is possible.

Method 6: Priority Replanning: Algorithmic / Computational Improvement

As both the CSP and MPP algorithms still retain the capability to be halted after a period of time has elapsed, their *anytime* property coupled with the *plan repair strategy* opens up the opportunity for replanning to be prioritised based on the influence of the ROIs. Given that the order of the initial tour is maintained, it is easy to determine which segments of the plan require immediate resolution. Replanning times could be further minimised by resolving the ROIs that require immediate attention. Distant ROIs can be resolved later, if time permits. Distant ROIs do not have to be replanned immediately as it can be assumed that those regions may change later during the inspection.

As discussed in Section 2.2.3, hierarchical path planners HPA* and HAA* make the same prioritisation when replanning in real-time for video game applications (Botea et al., 2004; Harabor and Botea, 2008). The abstract graph is solved to give an indication of a possible path, but the actual path is solved incrementally as it is assumed that future changes in the environment will likely invalidate the current path at some point along the way to the goal. A parallel implementation will ensure *n*-ROIs could be replanned for each planning update, where *n* is the number of processors.

Method 7: Improving Coverage Quality by Using Quality Metrics: Algorithmic / Quality Improvement

Currently, calculating the minimal set cover is determined by a '*most primitives observed*' metric. Configurations with the highest primitive count do not guarantee the best quality in coverage. In *Chapter 7*, the *optimal sampling procedure* was proposed to increase the regularity of optimally placed configuration over random sampling to increase the quality of coverage in areas of simple geometry. For visual inspection tasks where image quality is important, it may be better to have two or more observations of a surface than one observation that contains the most primitives.

A more appropriate selection criterion may be to evaluate coverage based on the quality of the observation. An illumination metric such as *Lambert's Cosine Law* (Luna, 2012) may provide as a better indication to the quality of an observation rather than the quantity of the observation. Incorporating such a metric would increase the sizes of the covering sets and consequently produce longer inspection plans. However, if implemented correctly would ensure a more high-quality visual inspection is performed.

Method 8: Path Smoothing over Sub-MPPs: Quality Improvement

To improve the *optimality* of the tour, the post-optimisation path smoothing algorithm developed by Englot and Hover (2012a) can be applied over each sub-MPP and the larger areas surrounding the ROIs to smooth the transition between existing and newly replanned areas. The smoothing technique iteratively examines each configuration of the tour and attempts smooth out the path by sampling a new configuration pushing it towards the *optimal cost frontier* that connects neighbouring configurations in the tour. The result of the path smoothing algorithm process reduced the length of the tour by minimising travel distance between configurations and removed any configurations that no longer unique primitives.

If an *anytime* approach to replanning is taken (*Method 5*) and there is time remaining before the plan update is required, the remaining time can be allocated to improve upon the quality of recently solved sub-MPPs. If the prioritised replanning approach is taken (*Method 6*), lower prioritised regions can be iteratively updated until the paths segments are required to be traversed by the robot. These approaches should be achievable as the online implementation of the *offline sampling-based coverage planner* developed by Song and Jo (2018) utilised the path smoothing procedure in real time.

Method 9: Implement Adaptive Sampling-based Coverage Planner on a Physical Platform: Algorithmic / Computational Improvement

The final suggestion of future work is aimed at what this thesis was not able to achieve; the real-world testing of the *adaptive sampling-based coverage planner* on a physical multi-legged platform (*Limitation 5*). At the time of the scheduled physically trials, the *adaptive motion planner* had not been successfully integrated for use on the concept demonstrator. This limited the capability of the platform to move autonomously and motion planner that cycled between pre-calculated gait cycles was used as an ad-hock replacement. However, given the complexities of autonomously navigating a multi-legged platform, it was not sufficient for the physical trials.

The modification of the PhantomX platform to create the concept demonstrator created unforeseen issues that hindered the physical trials. An intermittent issue would arise where one of the servomotors would stall a leg of the robot, rendering it unable to move. A hard reset of the robot restored the motion to the robot, at the expensive of terminating the inspection, but as the fault was not repeatable it was difficult to determine the cause. By the time of completing this thesis, the source of the problem had still not been determined or resolved.

Despite the issues with the platform, the *mapping system* as proposed in Pivetta et al. (2017) was successfully integrated with *adaptive coverage planner*. Proof of concept trials highlighted the *mapping system* and *inspection planner modules* worked together as expected under the assumptions that were placed in this thesis. However, the inability of the concept demonstrator to successfully complete an inspection of the testing environment produced insufficient data to present conclusive results, so these preliminary results were excluded from the thesis.

If the concept demonstrator were to be operational, the next phase of the IPF development would have been to extend the capability of both the *mapping system* and *adaptive coverage planner* to account for positional and sensor uncertainty (*Assumption 5*), which is imperative for robust online execution of an inspection plan. Instead of the *mapping system* just providing a map update to the *inspection planner*, further information would need to be supplied about the uncertainty of the mesh model to ensure that preserved and new viewing locations observe their intended coverage. For the *adaptive coverage planner*, these cross checks will have to be performed during the ROI Validation phase for all configurations.

For configurations that lose coverage due to positional and sensor error, the potentially unseen primitives could be deemed unobserved and can have their coverage revaluated during the Online CSP phase. Newly created viewing locations can have the uncertainty or error accounted for at the time of calculation.

To be verify that the coverage is achieved by each configuration given positional and sensor uncertainty, the *executive* (Figure 1-6) that governs the IPF must cross check the actual coverage taken by the robot against the expected coverage listed under that configuration. If coverage is missing the *executive* could call upon the *inspection planning module* to generate new positions that cover the missing coverage without having to trigger a replan update. This would create an *inspection planning module* that is flexible to update its plan at any time during the inspection and not just when the mapping system detects new information about the environment. Furthermore, if there are any discrepancies in coverage, it can be used by the *mapping system module* (Pivetta et al., 2017) to improve the positional estimates of the robot.

Finally, to get the *adaptive coverage planner* working on a multi-legged platform in submarine tank-like environments, it will require a platform that possesses magnetic adhesion. This will require a redesign of the current concept demonstrator to accommodate the additional components. Given the algorithmic capability developed, the robot with magnetic adhesion should be possible to deploy and test the platform in a real-world submarine tank or tank-like environment. Additionally, a redesign with fewer legs on the platform may assist in reducing the computational expense of planning multi-legged platforms.

9.5 Concluding Statement

This thesis investigated the *submarine tank inspection planning problem* and in doing so provided insight into improving existing coverage planning methods for the application of automated confined space inspection. To resolve this problem, an *adaptive sampling-based coverage planner* was proposed as an extension to a known offline coverage planner using a *plan replan strategy*. The results revealed that implementing a *plan repair strategy*, that isolated preserved segments from being replanned, resulted in a dramatic reduction in the time taken to replanning new features within the environments. Further, the tour quality incurred negligible negative effects, and in some cases, there were improvements.

A computational analysis was performed for the offline coverage algorithm, and in the process of performing this analysis, several limitations in the offline approach were discovered when applied to problems of this scale. This resulted in the development of new heuristics, in the form of *additional termination conditions* and the *hybrid-heuristic*, that dramatically improved upon the performance of current techniques. The application of these heuristics improved both offline and online coverage planning approaches, as measured by the time taken to complete the plan.

To conclude, while further work is required to validate this *adaptive sampling-based coverage planner* in a real-world scenario, the analysis performed herein suggests that the new *adaptive coverage planner* has the potential to enable a multi-legged robot to conduct the required inspection task. This work takes the next step towards high-quality automated submarine tank inspection to support and advance the sustainability of the Australian Collins Class submarine fleet.

Appendix A

Software Packages used to Create Coverage Planners and Heuristics

Table A-1 provides a list of software packages used to create the algorithms in this thesis.

Table A-1: The software and resources used to construct and visualise the results of the algorithms developed in this thesis.

Offline / Adaptive Online Coverage Planner		
Software Package	**Use**	**Source**
AVL Tree	Storing the mesh structure and associated primitive lists: *observed*, *unobserved*, and *unobservable*	C++ implementation of java code derived from tutorials at: https://algorithms.tutorialhorizon.com/avl-tree-insertion/ and https://www.geeksforgeeks.org/avl-tree-set-2-deletion/
Optimal Collision Detection (OPCODE)	Collision checking	OPCODE: http://www.codercorner.com/Opcode.htm Used implementation from Open Dynamics Engine https://www.ode.org/
Point Cloud Library (PCL) and Fast Library for Approximate Nearest Neighbours (FLANN)	PCL provided the interface to use FLANN to perform nearest neighbour queries.	PCL: http://pointclouds.org/ FLANN: https://github.com/mariusmuja/flann
Concorde	Quick-Boruka and Chained Lin Kernighan Heuristic to solve TSP	http://www.math.uwaterloo.ca/tsp/concorde.html
Open Motion Planning Library (OMPL)	RRT-Connect	https://ompl.kavrakilab.org/
OpenSceneGraph (OSG)	Ray tracing	http://www.openscenegraph.org/

Boost (uBLAS)	Matrix data structure to multiply rotations from the spherical to Cartesian frame	https://www.boost.org/doc/libs/1_65_0/libs/numeric/ublas/doc/index.html
Environment Model Construction and Visualisation		
Software Package	**Use**	**Source**
Autodesk Inventor	Used to create environment models (STL)	https://www.autodesk.com.au/
MeshMesh	View and edit STL models to remove outer mesh layer	http://www.meshlab.net/
MATLAB 2018a	Data analysis and visualisation of results	https://au.mathworks.com/
ParaView	To view skeletons generated in the form of VTK files.	https://www.paraview.org/
Skeletisation		
BinVox	Convert STL files in to a voxlised format for thinning.	https://www.patrickmin.com/binvox/
Insight Toolkit (ITK)	Thinning algorithm for the *hybrid-huersitc.* An implementation of Homman's implementation of Lee's 3D thinning algorithm. Exports VTK files	https://itk.org/ (BinaryThinningImageFilter)

Bibliography

Acar, E. U. and Choset, H. (2002). Sensor-based coverage of unknown environments: Incremental construction of morse decompositions. *The International Journal of Robotics Research*, 21(4):345–366.

Acar, E. U., Choset, H., Rizzi, A. A., Atkar, P. N., & Hull, D. (2002). Morse decompositions for coverage tasks. *The international journal of robotics research*, *21*(4), 331-344.

Acar, E. U., Choset, H., Zhang, Y., & Schervish, M. (2003). Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods. *The International journal of robotics research*, *22*(7-8), pp. 441-466.

Alatartsev, S., Stellmacher, S., & Ortmeier, F. (2015). Robotic task sequencing problem: A survey. *Journal of intelligent & robotic systems*, *80*(2), 279-298.

Almadhoun, R., Taha, T., Seneviratne, L., Dias, J., & Cai, G. (2016). A survey on inspecting structures using robotic systems. *International Journal of Advanced Robotic Systems*, *13*(6), 1729881416663664.

Applegate, D. L., R. E. Bixby, V. Chvatal and W. J. Cook (2011). The Traveling Salesman Problem: A Computational Study: A Computational Study, Princeton University Press.

Applegate, D., Cook, W., & Rohe, A. (2003). Chained Lin-Kernighan for large traveling salesman problems. *INFORMS Journal on Computing*, *15*(1), 82-92.

Arslan, O., & Tsiotras, P. (2013). Use of relaxation methods in sampling-based algorithms for optimal motion planning. In *2013 IEEE International Conference on Robotics and Automation* (pp. 2421-2428). IEEE.

ASC Pty Ltd. (n.d). On board a submarine [PDF]. Available at: https://www.asc.com.au/assets/downloads/Resources/On_Board_a_Submarine.pdf. Accessed 14 December 2019.

Atkar, P. N., Conner, D. C., Greenfield, A., Choset, H., & Rizzi, A. A. (2008). Hierarchical segmentation of piecewise pseudoextruded surfaces for uniform coverage. *IEEE Transactions on Automation Science and Engineering*, *6*(1), 107-120.

Atkar, P. N., Greenfield, A., Conner, D. C., Choset, H., & Rizzi, A. A. (2005). Uniform coverage of automotive surface patches. *The International Journal of Robotics Research*, *24*(11), pp. 883-898.

Australian Government: Department of Defence. (2016). 2016 Defence White Paper. *Commonwealth of Australia*. Available at: https://www.defence.gov.au/WhitePaper/Docs/2016-Defence-White-Paper.pdf, Accessed 23 December 2019

Australian Government: Department of Industry, Innovation and Science. (2019). *Industry 4.0*. [online] Available at: https://www.industry.gov.au/funding-and-incentives/manufacturing/industry-40, Accessed 30 September. 2019.

Bajcsy, R. (1988). Active perception. *Proceedings of the IEEE, 76*(8), pp. 966-1005.

Bakken, R. H., & Eliassen, L. M. (2017). Real-time three-dimensional skeletonisation using general-purpose computing on graphics processing units applied to computer vision-based human pose estimation. *The International Journal of High Performance Computing Applications*, *31*(4), 259-273.

Banta, JE, Wong, LR, Dumont, C, & Abidi, MA. (2000). A next-best-view system for autonomous 3-D object reconstruction. *IEEE Transactions on Systems, Man, and Cybernetics. a Publication of the IEEE Systems, Man, and Cybernetics Society.*, *30*(5), 589-598

Bertola, A, Gonzalez, LF (2013). Adaptive dynamic path re-planning RRT algorithms with game theory for UAVs. In: 15th Australian international aerospace congress (AIAC‘15), Melbourne, Australia, pp. 613–626

Bhattacharya, S. (2010, July). Search-based path planning with homotopy class constraints. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Bibuli, M., Bruzzone, G., Caccia, M., Ortiz, A., Vögele, T., Eich, M. & Vergine, A. (2011). The MINOAS project: Marine INspection Robotic Assistant System. In *2011 19th Mediterranean Conference on Control & Automation (MED)*, pp. 1188-1193. IEEE.

Bircher, A., Alexis, K., Burri, M., Oettershagen, P., Omari, S., Mantel, T., & Siegwart, R. (2015). Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics. In *2015 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 6423-6430). IEEE.

Bircher, A., Alexis, K., Schwesinger, U., Omari, S., Burri, M., & Siegwart, R. (2017). An incremental sampling-based approach to inspection planning: the rapidly exploring random tree of trees. *Robotica*, *35*(6), 1327-1340.

Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., & Siegwart, R. (2016). Receding horizon" next-best-view" planner for 3d exploration. In *2016 IEEE international conference on robotics and automation (ICRA)* (pp. 1462-1468). IEEE.

Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., & Siegwart, R. (2018). Receding horizon path planning for 3D exploration and surface inspection. *Autonomous Robots*, *42*(2), 291-306.

Blaer, P. S., & Allen, P. K. (2009). View planning and automated data acquisition for three-dimensional modeling of complex sites. *Journal of Field Robotics*, *26*(11-12), 865-891.

Blum, H. (1967). A transformation for extracting new descriptors of shape. *Models for the perception of speech and visual form*, *19*(5), pp. 362-380.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., & Zhang, X. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.

Borgerink, D. J., Stegenga, J., Brouwer, D. M., Wörtche, H. J. and Stramigioli, S. (2014). Rail-guided robotic end-effector position error due to rail compliance and ship motion. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3463-3468.

Botea, A., Bouzy, B., Buro, M., Bauckhage, C., & Nau, D. (2013). Pathfinding in Games. *Artificial and Computational Intelligence in Games*, pp. 21-31.

Botea, A., Müller, M., & Schaeffer, J. (2004). Near optimal hierarchical path-finding. *Journal of game development*, *1*(1), 7-28.

Botti, L., Ferrari, E., & Mora, C. (2017). Automated entry technologies for confined space work activities: A survey. *Journal of occupational and environmental hygiene*, *14*(4), 271-284.

Bruce, J., & Veloso, M. M. (2002). Real-time randomized path planning for robot navigation. In *Robot Soccer World Cup* (pp. 288-295). Springer, Berlin, Heidelberg.

Bruck, J., Gao, J., & Jiang, A. (2007). MAP: Medial axis based geometric routing in sensor networks. *Wireless Networks*, *13*(6), pp. 835-853.

Brunner, M., Brüggemann, B., & Schulz, D. (2013). Hierarchical rough terrain motion planning using an optimal sampling-based method. In *2013 IEEE International Conference on Robotics and Automation* (pp. 5539-5544). IEEE.

Brusell, A., Andrikopoulos, G., & Nikolakopoulos, G. (2016). A survey on pneumatic wall-climbing robots for inspection. In *2016 24th Mediterranean Conference on Control and Automation (MED)* (pp. 220-225). IEEE.

Buniyamin, N., Ngah, W. W., Sariff, N., & Mohamad, Z. (2011). A simple local path planning algorithm for autonomous mobile robots. *International journal of systems applications, Engineering & development*, *5*(2), 151-159.

Burton Jr, R. M., & Mpitsos, G. J. (1992). Event-dependent control of noise enhances learning in neural networks. *Neural Networks*, *5*(4), 627-637.

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., & Leonard, J. J. (2016). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, *32*(6), pp. 1309-1332.

Cao, Z. L., Huang, Y., & Hall, E. L. (1988). Region filling operations with random obstacle avoidance for mobile robots. *Journal of Robotic systems*, *5*(2), pp. 87-102.

Carreras, M., Hernández, J. D., Vidal, E., Palomeras, N., Ribas, D., & Ridao, P. (2018). Sparus II AUV—a hovering vehicle for seabed inspection. *IEEE Journal of Oceanic Engineering*, *43*(2), 344-355.

Carsten, J., Ferguson, D., & Stentz, A. (2006, October). 3d Field D: Improved path planning and replanning in three dimensions. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 3381-3386). IEEE.

Chen, S. Y., & Li, Y. F. (2004). Automatic sensor placement for model-based robot vision. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *34*(1), 393-408.

Chen, S., Li, Y., & Kwok, N. M. (2011). Active vision in robotic systems: A survey of recent developments. *The International Journal of Robotics Research*, *30*(11), 1343-1377.

Cheng, P., Keller, J., & Kumar, V. (2008). Time-optimal UAV trajectory planning for 3D urban structure coverage. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2750-2757. IEEE.

Chew, L. P. (1989). Constrained delaunay triangulations. *Algorithmica*, *4*(1-4), 97-108.

Chin, W. P., & Ntafos, S. (1986, August). Optimum watchman routes. In *Proceedings of the second annual symposium on Computational geometry* (pp. 24-33).

Choset, H. (2000). Coverage of known spaces: The boustrophedon cellular decomposition. *Autonomous Robots*, *9*(3), 247-253.

Choset, H. (2001). Coverage for robotics–A survey of recent results. *Annals of mathematics and artificial intelligence*, *31*(1-4), pp. 113-126.

Choset, H. M., Hutchinson, S., Lynch, K. M., Kantor, G., Burgard, W., Kavraki, L. E., & Thrun, S. (2005). *Principles of robot motion: theory, algorithms, and implementation*. MIT press.

Choset, H., & Burdick, J. (2000). Sensor-based exploration: The hierarchical generalized voronoi graph. *The International Journal of Robotics Research*, *19*(2), pp. 96-125.

Choset, H., & Pignon, P. (1998). Coverage path planning: The boustrophedon cellular decomposition. In *Field and service robotics* (pp. 203-209). Springer, London.

Choset, H., Acar, E., Rizzi, A. A., & Luntz, J. (2000, April). Exact cellular decompositions in terms of critical points of morse functions. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)* (Vol. 3, pp. 2270-2277). IEEE.

Christensen, L., Fischer, N., Kroffke, S., Lemburg, J., & Ahlers, R. (2011a). Cost-effective autonomous robots for ballast water tank inspection. *Journal of ship production and design*, *27*(3), pp. 127-136.

Christensen, L., Lemburg, J., Vögele, T., Kirchner, F., Fischer, N., Ahlers, R., & Etzold, L. E. (2011b). Tank inspection by cost effective rail based robots. In *ICCAS*. Trieste, Italy.

Christofides, N. (1976). Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report, CS-93-13, Carnegie-Mellon University.

Chvatal, V. (1979). A greedy heuristic for the set-covering problem. *Mathematics of operations research*, *4*(3), 233-235.

Cohen, J. (2013). Statistical power analysis for the behavioral sciences. Routledge.

Coles, J., Greenfield, P., Spark, M., & Savage, H. (2016). Study into the business of sustaining Australia's strategic Collins class submarine capability–Beyond benchmark. *Canberra: Commonwealth of Australia*.

Connolly, C. (1985). The determination of next best views. Proceedings. 1985 IEEE International Conference on Robotics and Automation, 2, 432-435.

Cornea, N. D., Silver, D., & Min, P. (2007). Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on Visualization & Computer Graphics*, (3), pp. 530-548.

CSIRO - DATA 61: The Robotics and Autonomous Systems Group. Magnapods: Highly Flexible Inspection Robots (2016). Cited 24/8/2017. Available at: https://research.csiro.au/robotics/magnapods/

Cui, X., & Shi, H. (2011). A*-based pathfinding in modern computer games. *International Journal of Computer Science and Network Security*, *11*(1), 125-130

Current, J. R., & Schilling, D. A. (1989). The covering salesman problem. *Transportation science*, 23(3), 208-213.

Dakulovic, M., & Petrovic, I. (2012). Complete coverage path planning of mobile robots for humanitarian demining. *Industrial Robot: An International Journal*, *39*(5), pp. 484-493.

Danner, T., & Kavraki, L. E. (2000). Randomized planning for short inspection paths. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation.* Volume 2, pp. 971-976. IEEE.

De Baere, K., Verstraelen, H., Rigo, P., Van Passel, S., Lenaerts, S., & Potters, G. (2013). Reducing the cost of ballast tank corrosion: an economic modeling approach. *Marine Structures*, *32*, pp. 136-152.

Demyen, D., & Buro, M. (2006, July). Efficient triangulation-based pathfinding. In A*aai* (Vol. 6, pp. 942-947).

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, *1*(1), 269-271.

Dong, X., Palmer, D., Axinte, D., & Kell, J. (2019). In-situ repair/maintenance with a continuum robotic machine tool in confined space. *Journal of Manufacturing Processes*, *38*, 313-318.

Dorigo, M., & Birattari, M. (2010). *Ant colony optimization* (pp. 36-39). Springer US.

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, *1*(1), 53-66.

Dubins, L. E. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, *79*(3), 497-516.

Elbanhawi, M., & Simic, M. (2014). Sampling-based robot motion planning: A review. *IEEE Access*, *2*, 56-77.

Ellefsen, K. O., Lepikson, H. A., & Albiez, J. C. (2017). Multiobjective coverage path planning: Enabling automated inspection of complex, real-world structures. *Applied Soft Computing*, *61*, 264-282.

Englot, B., & Hover, F. (2012a). Sampling-based coverage path planning for inspection of complex structures. In *Twenty-Second International Conference on Automated Planning and Scheduling*.

Englot, B., & Hover, F. (2012b). Sampling-based sweep planning to exploit local planarity in the inspection of complex 3D structures. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4456-4463. IEEE.

Englot, B., & Hover, F. (2013). Three-dimensional coverage planning for an underwater inspection robot. *The International Journal of Robotics Research*, *32*(9-10), pp. 1048-1073.

Englot, B., & Hover, F. (2017). Planning complex inspection tasks using redundant roadmaps. In *Robotics Research* (pp. 327-343). Springer, Cham.

Englot, B., (2012). Sampling-based coverage path planning for complex 3D structures (Doctoral dissertation), *Massachusetts Institute of Technology*. http://hdl.handle.net/1721.1/78209

Englot, B., and Hover, F. (2011). Planning complex inspection tasks using redundant roadmaps. *Proceedings. International Symposium Robotics Research*.

Ferguson, D., & Stentz, A. (2005). The Field D* algorithm for improved path planning and replanning in uniform and non-uniform cost environments. *Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-19*.

Ferguson, D., & Stentz, A. (2007). Anytime, dynamic planning in high-dimensional search spaces. In *Proceedings 2007 IEEE International Conference on Robotics and Automation* (pp. 1310-1315). IEEE.

Ferguson, D., Kalra, N., & Stentz, A. (2006). Replanning with RRTs. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* (pp. 1243-

1248). IEEE.

Floyd, R. W. (1962). Algorithm 97: shortest path. *Communications of the ACM*, *5*(6), 345. doi: 10.1145/367766.368168

Fortune, S. (1995). Voronoi diagrams and Delaunay triangulations. In *Computing in Euclidean geometry* (pp. 225-265).

Gabriely, Y., & Rimon, E. (2002). Spiral-STC: An on-line coverage algorithm of grid environments by a mobile robot. In *Proceedings - 2002 IEEE International Conference on Robotics and Automation*, Vol. 1, pp. 954-960.

Galceran, E., & Carreras, M. (2013). A survey on coverage path planning for robotics. *Robotics and Autonomous systems*, *61*(12), pp. 1258-1276.

Galceran, E., Campos, R., Palomeras, N., Ribas, D., Carreras, M., & Ridao, P. (2015). Coverage path planning with real-time replanning and surface reconstruction for inspection of three-dimensional underwater structures using autonomous underwater vehicles. *Journal of Field Robotics*, *32*(7), pp. 952-983.

Gammell, J. D., Srinivasa, S. S., & Barfoot, T. D. (2014). Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2997-3004). IEEE.

González-Baños, H. and Latombe, J. C. (1998). Planning robot motions for range image acquisition and automatic 3D model construction. In *Proceedings of the AAAI Fall Symposium Series, Integrated Planning for Autonomous Agent Architectures*, pp. 23-25

González-Baños, H. and Latombe, J. C. (2001). A randomized art-gallery algorithm for sensor placement. In *Proceedings of the 17th Annual Symposium on Computational geometry*, pp. 232-240). ACM.

González-Baños, H., & Latombe, J. (2002). Navigation Strategies for Exploring Indoor Environments. *The International Journal of Robotics Research, 21*(10-11), 829-848.

Graham, R. L., & Hell, P. (1985). On the history of the minimum spanning tree problem. *Annals of the History of Computing*, *7*(1), 43-57.

Gu, Y., Li, N., Zhang, F., & Li, L. (2009). Corrosion and protection of submarine metal components in seawater. In *2009 8th International Conference on Reliability, Maintainability and Safety*, pp. 1226-1229. IEEE.

Haines, E. (1994). Point in polygon strategies. *Graphics gems IV*, *994*, 24-26.

Hameed, I. A. (2014). Intelligent coverage path planning for agricultural robots and autonomous machines on three-dimensional terrain. *Journal of Intelligent & Robotic Systems*, *74*(3-4), pp. 965-983.

Harabor, D., & Botea, A. (2008). Hierarchical path planning for multi-size agents in heterogeneous environments. In *2008 IEEE Symposium on Computational Intelligence and Games* (pp. 258-265). IEEE.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, *4*(2), 100-107

Haussler, D., & Welzl, E. (1987). ε-nets and simplex range queries. *Discrete & Computational Geometry*, *2*(2), 127-151.

Helsgaun, K. (2000). An effective implementation of the Lin–Kernighan traveling salesman heuristic. *European Journal of Operational Research*, *126*(1), pp. 106-130.

Heng, L., Gotovos, A., Krause, A., & Pollefeys, M. (2015). Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments. In *2015 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1071-1078). IEEE.

Hernández, J. D., Vidal, E., Greer, J., Fiasco, R., Jaussaud, P., Carreras, M., & García, R. (2017). AUV online mission replanning for gap filling and target inspection. In *OCEANS 2017-Aberdeen* (pp. 1-4). IEEE.

Hernández, J. D., Vidal, E., Moll, M., Palomeras, N., Carreras, M., & Kavraki, L. E. (2019). Online motion planning for unexplored underwater environments using autonomous underwater vehicles. *Journal of Field Robotics*, *36*(2), 370-396.

Heyer, A., D'Souza, F., Morales, C. L., Ferrari, G., Mol, J. M. C., & De Wit, J. H. W. (2013). Ship ballast tanks a review from microbial corrosion and electrochemical point of

view. *Ocean Engineering*, *70*, pp. 188-200.

Homann, H. (2007). Implementation of a 3D thinning algorithm. *Insight Journal*, (July – December)

Hörger, M. (2014). *Real-time stabilisation for hexapod robots using task-space constraints* (Master's thesis).

Hover, F. S., Eustice, R. M., Kim, A., Englot, B., Johannsson, H., Kaess, M., and Leonard, J. J. (2012). Advanced perception, navigation and planning for autonomous in-water ship hull inspection. *The International Journal of Robotics Research*, 31(12):1445–1464.

How, J. P., Behihke, B., Frank, A., Dale, D., & Vian, J. (2008). Real-time indoor autonomous vehicle test environment. *IEEE Control Systems Magazine*, *28*(2), 51-64.

Islam, F., Nasir, J., Malik, U., Ayaz, Y., & Hasan, O. (2012). RRT*-Smart: Rapid convergence implementation of RRT* towards optimal solution. In *2012 IEEE International Conference on Mechatronics and Automation* (pp. 1651-1656). IEEE.

Isler, V., Kannan, S., & Daniilidis, K. (2004). Sampling based sensor-network deployment. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)* (Vol. 2, pp. 1780-1785). IEEE.

Jalba, A. C., Kustra, J., & Telea, A. C. (2012). Surface and curve skeletonization of large 3D models on the GPU. *IEEE transactions on pattern analysis and machine intelligence*, *35*(6), 1495-1508.

Janglová, D. (2004). Neural networks in mobile robot motion. *International Journal of Advanced Robotic Systems*, *1*(1), 2.

Jin, J., & Tang, L. (2011). Coverage path planning on three-dimensional terrain for arable farming. *Journal of field robotics*, *28*(3), pp. 424-440.

Jing, W., Polden, J., Goh, C. F., Rajaraman, M., Lin, W., & Shimada, K. (2017b). Sampling-based coverage motion planning for industrial inspection application with redundant robotic system. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 5211-5218). IEEE.

Jing, W., Polden, J., Lin, W., & Shimada, K. (2016). Sampling-based view planning for 3d visual coverage task with unmanned aerial vehicle. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1808-1815). IEEE.

Jing, W., Polden, J., Tao, P. Y., Goh, C. F., Lin, W., & Shimada, K. (2017a). Model-based coverage motion planning for industrial 3D shape inspection applications. In *2017 13th IEEE Conference on Automation Science and Engineering (CASE)* (pp. 1293-1300). IEEE

Johnson, G. (2006). Smoothing a Navigation Mesh Path. In *AI Game Programming Wisdom 3*, Charles River Media, pp.134-135

Johnson, H. J., McCormick, M., & Ibáñez, L. (2013). Consortium, TIS, The ITK Software Guide. Kitware. *Inc.*, In press. URL: http://www.itk.org/ItkSoftwareGuide.pdf.

Juliá, M., Gil, A., & Reinoso, O. (2012). A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots*, *33*(4), 427-444.

Kafka, P., Faigl, J., & Váňa, P. (2016). Random Inspection Tree Algorithm in visual inspection with a realistic sensing model and differential constraints. In *2016 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 2782-2787). IEEE.

Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., & Schaal, S. (2011). STOMP: Stochastic trajectory optimization for motion planning. In *2011 IEEE international conference on robotics and automation* (pp. 4569-4574). IEEE.

Kallmann, M. (2005). Path planning in triangulations. In Proceedings of the IJCAI workshop on reasoning, representation, and learning in computer games (pp. 49-54).

Kalra, L. P., Gu, J., & Meng, M. (2006). A wall climbing robot for oil tank inspection. In *2006 IEEE International Conference on Robotics and Biomimetics* (pp. 1523-1528). IEEE.

Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, *30*(7), 846-894.

Karaman, S., Walter, M. R., Perez, A., Frazzoli, E., & Teller, S. (2011). Anytime motion planning using the RRT. In *2011 IEEE International Conference on Robotics and*

Automation (pp. 1478-1483). IEEE.

Katrasnik, J., Pernus, F., & Likar, B. (2009). A survey of mobile robots for distribution power line inspection. *IEEE Transactions on power delivery*, *25*(1), pp. 485-493.

Kavraki, L. E., Kolountzakis, M. N., & Latombe, J. C. (1998). Analysis of probabilistic roadmaps for path planning. *IEEE Transactions on robotics and automation*, *14*(1), 166-171.

Kavraki, L. E., Svestka, P., Latombe, J. C., & Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, *12*(4), 566-580.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous robot vehicles* (pp. 396-404). Springer, New York, NY.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, *220*(4598), 671-680.

Klingensmith, M., Dryanovski, I., Srinivasa, S., & Xiao, J. (2015, July). Chisel: Real Time Large Scale 3D Reconstruction Onboard a Mobile Device using Spatially Hashed Signed Distance Fields. In *Robotics: science and systems* (Vol. 4, p. 1).

Koenig, S., & Likhachev, M. (2002). D* Lite. *Aaai/iaai*, *15*.

Koenig, S., Likhachev, M., & Furcy, D. (2004). Lifelong planning A*. *Artificial Intelligence*, *155*(1-2), 93-146.

Koenig, S., Likhachev, M., Liu, Y., & Furcy, D. (2004). Incremental heuristic search in AI. *AI Magazine*, *25*(2), 99-99.

Kuffner, J. J., & LaValle, S. M. (2000). RRT-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings* (Cat. No. 00CH37065) (Vol. 2, pp. 995-1001). IEEE.

Laporte, G. (1992). The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, *59*(2), 231-247.

Larsen, E. (1999). Collision detection for real-time simulation. *Case study, Research and Development Department, SCEA*.

Latombe, J. C. (2012). *Robot motion planning* (Vol. 124). Springer Science & Business Media.

LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning.

LaValle, S. M. (2006). *Planning algorithms*. Cambridge university press.

LaValle, S. M., & Kuffner Jr, J. J. (2001). Randomized kinodynamic planning. *The international journal of robotics research, 20*(5), 378-400.

Lee, D. T., & Preparata, F. P. (1984). Euclidean shortest paths in the presence of rectilinear barriers. *Networks, 14*(3), 393-410.

Lee, D., Ku, N., Kim, T. W., Lee, K. Y., Kim, J., & Kim, S. (2010). Self-traveling robotic system for autonomous abrasive blast cleaning in double-hulled structures of ships. *Automation in Construction, 19*(8), pp. 1076-1086.

Lee, T. C., Kashyap, R. L., & Chu, C. N. (1994). Building skeleton models via 3-D medial surface axis thinning algorithms. *CVGIP: Graphical Models and Image Processing*, *56*(6), pp. 462-478.

Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D., Kammel, S., ... & Sokolsky, M. (2011). Towards fully autonomous driving: Systems and algorithms. In *2011 IEEE Intelligent Vehicles Symposium (IV)* (pp. 163-168). IEEE.

Lin, S., & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, *21*(2), 498-516.

Liu, B., Telea, A. C., Roerdink, J. B., Clapworthy, G. J., Williams, D., Yang, P., & Chiarini, A. (2014). Parallel centerline extraction on the GPU. *Computers & Graphics*, *41*, 72-83.

Liu, Y., Lin, X., & Zhu, S. (2008). Combined coverage path planning for autonomous cleaning robots in unstructured environments. In *2008 7th World Congress on Intelligent Control and Automation*, pp. 8271-8276. IEEE.

Lozano-Pérez, T., & Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, *22*(10), 560-570.

Luna, F. (2012). Introduction to 3D game programming with DirectX 11. *Stylus Publishing*, LLC.

Mac, T. T., Copot, C., Tran, D. T., & De Keyser, R. (2016). Heuristic approaches in robot path planning: A survey. *Robotics and Autonomous Systems*, *86*, 13-28.

Mangan, A. P., & Whitaker, R. T. (1999). Partitioning 3D surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics*, *5*(4), 308-321.

Marden, P., & Smith, F. (2014). Dynamically updating a navigation mesh via efficient polygon subdivision. *Game Programming Wisdom 4*.

Meagher, D. (1982). Geometric modeling using octree encoding. *Computer graphics and image processing*, *19*(2), 129-147.

Mills, G. H., Liu, J. H., Kaddouh, B. Y., Jackson, A. E., & Richardson, R. C. (2018). Miniature Magnetic Robots for In-Pipe Locomotion. In *Robotics Transforming the Future: Proceedings of CLAWAR 2018: The 21st International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines* (pp. 289-300). CLAWAR Association Ltd.

Milnor, J. (2016).*Morse theory. (AM-51)* (Vol. 51). Princeton university press.

Min, P. (2017). Binvox 3D mesh voxelizer. *Available on: http://www.patrickmin.com/binvox/*, accessed March 2017.

Mitchell, M. (1998). An introduction to genetic algorithms. MIT press.

Montero, R., Victores, J. G., Martinez, S., Jardón, A., & Balaguer, C. (2015). Past, present and future of robotic tunnel inspection. *Automation in Construction*, *59*, pp. 99-112.

Muja, M., & Lowe, D. G. (2009). Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, *2*, pp. 331-340.

Nooruddin, F. S., & Turk, G. (2003). Simplification and repair of polygonal models using volumetric techniques. *IEEE Transactions on Visualization and Computer Graphics*, *9*(2),

pp. 191-205.

Noreen, I., Khan, A., & Habib, Z. (2016). A comparison of RRT, RRT* and RRT*-smart path planning algorithms. *International Journal of Computer Science and Network Security (IJCSNS)*, *16*(10), 20.

Nunes, E., Peixoto, A., Xaud, M., From, P. R., Carvalho, G. H., Oliveira, J. F., … Costa, R. (2013, October 29). DORIS - Monitoring Robot for Offshore Facilities. Offshore Technology Conference. *Congresso Brasileiro de Automática*. doi:10.4043/24386-MS.

Nykolaychuk, M., & Ortmeier, F. (2015). Coverage Path Re-planning for Processing Faults. In *International Conference on Intelligent Robotics and Applications* (pp. 358-368). Springer, Cham.

Orman, A. J., & Williams, H. P. (2007). A survey of different integer programming formulations of the travelling salesman problem. In *Optimisation, econometric and financial analysis* (pp. 91-104). Springer, Berlin, Heidelberg.

O'Rourke, J. (1987). *Art gallery theorems and algorithms* (Vol. 57). Oxford: Oxford University Press.

Otte, M., & Frazzoli, E. (2015). RRT^{X} Real-Time Motion Planning/Replanning for Environments with Unpredictable Obstacles. In *Algorithmic Foundations of Robotics XI* (pp. 461-478). Springer, Cham.

Otte, M., & Frazzoli, E. (2016). RRT^{X}: Asymptotically optimal single-query sampling-based motion planning with quick replanning. *The International Journal of Robotics Research*, *35*(7), 797-822.

Palágyi, K., & Kuba, A. (1999). A parallel 3D 12-subiteration thinning algorithm. *Graphical Models and Image Processing*, *61*(4), pp. 199-221.

Palomeras, N., Hurtós, N., Carreras, M., & Ridao, P. (2018). Autonomous mapping of underwater 3-D structures: From view planning to execution. *IEEE Robotics and Automation Letters*, *3*(3), 1965-1971.

Papachristos, C., Kamel, M., Popović, M., Khattak, S., Bircher, A., Oleynikova, H., & Siegwart, R. (2019). Autonomous exploration and inspection path planning for aerial robots

using the robot operating system. In *Robot Operating System (ROS)* (pp. 67-111). Springer, Cham.

Papadopoulos, G., Kurniawati, H., & Patrikalakis, N. M. (2013). Asymptotically optimal inspection planning using systems with differential constraints. In *2013 IEEE International Conference on Robotics and Automation* (pp. 4126-4133). IEEE.

Patil, S., Burgner, J., Webster, R. J., & Alterovitz, R. (2014). Needle steering in 3-D via rapid replanning. *IEEE Transactions on Robotics*, *30*(4), 853-864.

Patle, B. K., Pandey, A., Parhi, D. R. K., & Jagadeesh, A. (2019). A review: On path planning strategies for navigation of mobile robot. *Defence Technology*.

Pivetta R., Lammas, A., Short, A., Johanasson, M., Wardle, C., Summat, K. and Van Duin, S. (2017). Submarine Tank Inspection Robots, *4th SIA Submarine Science, Technology and Engineering Conference 2017* (SubSTEC4)

Pivetta, R., Lammas, A. & Sammut, K. (2018). *3D Adaptive Coverage Planning for Confined Space Inspection Robots*, *Technology and Science for the Ships of the Future: Proceedings of NAV 2018*, pp. 496-503, doi:10.3233/978-1-61499-870-9-496

Potvin, J. Y. (1996). Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, *63*(3), 337-370.

Puntambekar, A. A. (2009). *Data structures and algorithms*. Technical Publications.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., ... & Ng, A. Y. (2009). ROS: an open-source Robot Operating System. In *ICRA workshop on open source software* (Vol. 3, No. 3.2, p. 5).

Rabin, S. (2015). Game AI pro 2: collected wisdom of game AI professionals. AK Peters/CRC Press.

Rabin, S. (2020). *Game AI Pro 360: Guide to Movement and Pathfinding*. Boca Raton: CRC Press, https://doi.org/10.1201/9780429055096

Richardson, R., Whitehead, S., Ng, T. C., Hawass, Z., Pickering, A., Rhodes, S., .& Mayfield, W. (2013). The “Djedi” robot exploration of the southern shaft of the Queen's

chamber in the great Pyramid of Giza, Egypt. *Journal of Field Robotics*, *30*(3), 323-348.

Rodriguez, S., Thomas, S., Pearce, R., & Amato, N. M. (2008). RESAMPL: A region-sensitive adaptive motion planner. In *Algorithmic Foundation of Robotics VII* (pp. 285-300). Springer, Berlin, Heidelberg.

Russell, S. J., & Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Rusu, R. B., & Cousins, S. (2011). 3d is here: Point cloud library (pcl). In *2011 IEEE international conference on robotics and automation* (pp. 1-4). IEEE.

Sabre Autonomous Solutions. Mobile Blasting Robot. (2019, 21 October 2019) Available from: http://www.sabreautonomous.com.au/

Sadiku, M. N. (2014). *Elements of electromagnetics*. Oxford university press.

Safe Work Australia. (2017, 17 March 2017). Confined spaces. Retrieved from https://www.safeworkaustralia.gov.au/confined.

Saha, M., Roughgarden, T., Latombe, J. C., & Sánchez-Ante, G. (2006). Planning tours of robotic arms among partitioned goals. *The International Journal of Robotics Research*, *25*(3), pp. 207-223.

Saha, M., Sánchez-Ante, G., & Latombe, J. C. (2003). Planning multi-goal tours for robot arms. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)* (Vol. 3, pp. 3797-3803). IEEE.

Samet, H. (1984). The quadtree and related hierarchical data structures. *ACM Computing Surveys (CSUR)*, *16*(2), 187-260.

Schmittler, J., Pohl, D., Dahmen, T., Vogelgesang, C., & Slusallek, P. (2005). Realtime ray tracing for current and future games. In *ACM SIGGRAPH 2005 Courses* (p. 23). ACM.

Scott, W. R. (2009). Model-based view planning. *Machine Vision and Applications*, *20*(1), pp. 47-69.

Scott, W. R., Roth, G., & Rivest, J. F. (2003). View planning for automated three-dimensional object reconstruction and inspection. *ACM Computing Surveys (CSUR)*, *35*(1),

pp. 64-96.

Sehestedt, S., Paul, G., Rushton-Smith, D., & Liu, D. (2013). Prior-knowledge assisted fast 3d map building of structured environments for steel bridge maintenance. In *2013 IEEE International Conference on Automation Science and Engineering (CASE)* (pp. 1040-1046). IEEE.

Shermer, T. C. (1992). Recent results in art galleries (geometry). *Proceedings of the IEEE, 80*(9), 1384-1399.

Shimrat, M. (1962). Algorithm 112: position of point relative to polygon. *Communications of the ACM, 5*(8), 434.

Short, A., & Bandyopadhyay, T. (2017). Legged motion planning in complex three-dimensional environments. *IEEE Robotics and Automation Letters*, *3*(1), pp. 29-36.

Short, A., Pan, Z., Larkin, N., & van Duin, S. (2016). Recent progress on sampling based dynamic motion planning algorithms. In *2016 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)* (pp. 1305-1311). IEEE.

Shukla, A., & Karki, H. (2013). A review of robotics in onshore oil-gas industry. In *2013 IEEE International Conference on Mechatronics and Automation*, pp. 1153-1160. IEEE.

Silva, M. F., Machado, J. T., & Tar, J. K. (2008). A survey of technologies for climbing robots adhesion to surfaces. In *2008 IEEE International Conference on Computational Cybernetics*, pp. 127-132. IEEE.

Snook, G. (2000). Simplified 3D Movement and Pathfinding Using Navigation Meshes. In *Game Programming Gems*, Charles River Media, pp. 288–304.

Song, S., & Jo, S. (2017). Online inspection path planning for autonomous 3D modeling using a micro-aerial vehicle. In *2017 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 6217-6224). IEEE.

Song, S., & Jo, S. (2018). Surface-based exploration for autonomous 3D modeling. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1-8). IEEE.

Srivastava, S. S., Kumar, S., Garg, R. C., & Sen, P. (1969). Generalized traveling salesman

problem through n sets of nodes. *CORS journal*, *7*(2), 97.

Stentz, A. (1997). Optimal and efficient path planning for partially known environments. In *Intelligent Unmanned Ground Vehicles* (pp. 203-220). Springer, Boston, MA.

Sucan, I. A., Moll, M., & Kavraki, L. E. (2012). The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4), 72-82.

Sun, W., Patil, S., & Alterovitz, R. (2015). High-frequency replanning under uncertainty using parallel sampling-based motion planning. *IEEE Transactions on Robotics*, *31*(1), 104-116.

Tagliasacchi, A., Delame, T., Spagnuolo, M., Amenta, N., & Telea, A. (2016). 3d skeletons: A state-of-the-art report. In *Computer Graphics Forum*, Vol. 35, No. 2, pp. 573-597.

Terdiman, P. (2001). Memory-optimized bounding-volume hierarchies. *URL www. codercorner. com/Opcode. pdf*.

Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, *99*(1), pp. 21-71.

Thrun, S., Burgard, W., & Fox, D. (2005). Probabilistic robotics. 2005. *Massachusetts Institute of Technology, USA*.

Tonneau, S., Mansard, N., Park, C., Manocha, D., Multon, F., & Pettré, J. (2018). A reachability-based planner for sequences of acyclic contacts in cluttered environments. In *Robotics Research* (pp. 287-303). Springer, Cham.

Treadgold, N. K., & Gedeon, T. D. (1998). Simulated annealing and weight decay in adaptive learning: the SARPROP algorithm. *IEEE Transactions on Neural Networks*, *9*(4), 662-668.

Trenkel, S., Weller, R., & Zachmann, G. (2007). A benchmarking suite for static collision detection algorithms.

Trossen Robotics. (2019). PhantomX AX Metal Hexapod Mark III. Cited 21/10/2019; Available from: https://www.trossenrobotics.com/phantomx-ax-hexapod.aspx.

University of Bremen: German Research Centre for Artificial Intelligence - Robotics

Innovation Center. (n.d). Autonomous Railguided Tank Inspection System. Retrieved 19 December 2019. Available at: https://robotik.dfki-bremen.de/en/research/robot-systems/artis.html.

Vazirani, V. V. (2013). *Approximation algorithms*. Springer Science & Business Media.

Vermette, J. (2011). A survey of path-finding algorithms employing automatic hierarchical abstraction. *University of Windsor. Available in: http://richard.myweb.cs. uwindsor. ca/cs510/vermette survey. pdf*.

Wang, P., Krishnamurti, R., & Gupta, K. (2007, April). View planning problem with combined view and traveling cost. In *Proceedings 2007 IEEE International Conference on Robotics and Automation* (pp. 711-716). IEEE.

Wang, R., & Qian, X. (2010). *OpenSceneGraph 3.0: Beginner's guide*. Packt Publishing Ltd

Ward, P. K., Manamperi, P., Brooks, P., Mann, P., Kaluarachchi, W., Matkovic, L. & Liu, D. (2014). Climbing robot for steel bridge inspection: Design challenges. In *Austroads Bridge Conference*. ARRB Group.

Weidner, N., Rahman, S., Li, A. Q., & Rekleitis, I. (2017). Underwater cave mapping using stereo vision. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5709-5715. IEEE.

Wheare, J. (2018). Mission planning for field robots using symbolic planning and topology (Doctoral Dissertation). Flinders University, Adelaide, Australia.

Wheare, J., Lammas, A., & Sammut, K. (2019). Toward the Generation of Mission Plans for Operation of Autonomous Marine Vehicles in Confined Areas. *IEEE Journal of Oceanic Engineering*, *44*(2), 320-330.

Williams, K., & Burdick, J. (2006). Multi-robot boundary coverage with plan revision. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* (pp. 1716-1723). IEEE.

Wu, C., Dai, C., Gong, X., Liu, Y. J., Wang, J., Gu, X. D., & Wang, C. C. (2019). Energy-Efficient Coverage Path Planning for General Terrain Surfaces. *IEEE Robotics and Automation Letters*, *4*(3), 2584-2591.

Wu, Y., Noel, A., Kim, D. D., Youcef-Toumi, K., & Ben-Mansour, R. (2015). Design of a maneuverable swimming robot for in-pipe missions. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4864-4871. IEEE.

Wurll, C., & Henrich, D. (2001). Point-to-point and multi-goal path planning for industrial robots. *Journal of Robotic Systems*, *18*(8), 445-461.

Wzorek, M., Kvarnström, J., & Doherty, P. (2010). Choosing path replanning strategies for unmanned aircraft systems. In *Twentieth International Conference on Automated Planning and*

Yan, H., Wang, H., Chen, Y., & Dai, G. (2008). Path planning based on constrained Delaunay triangulation. In *2008 7th World Congress on Intelligent Control and Automation* (pp. 5168-5173). IEEE.

Yang, G., Hu, C., Meng, H., & Wang, S. Y. (2019). Constraint Path Planning for an Autonomous Wall Spray Coating Robot. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)* (pp. 2977-2983). IEEE.

Yang, L., Qi, J., Song, D., Xiao, J., Han, J., & Xia, Y. (2016). Survey of robot 3D path planning algorithms. *Journal of Control Science and Engineering*, *2016*.

Yao, P., Cai, Y., & Zhu, Q. (2019). Time-optimal trajectory generation for aerial coverage of urban building. *Aerospace Science and Technology*, *84*, pp. 387-398.

Yu, S. N., Zhou, R., Xia, J., & Che, J. (2015). Decomposition and coverage of multi-UAV cooperative search area. *Journal of Beijing University of Aeronautics and Astronautics*, *41*(1), pp. 167-173.

Zelinsky, A., Jarvis, R. A., Byrne, J. C., & Yuta, S. (1993). Planning paths of complete coverage of an unstructured environment by a mobile robot. In *Proceedings of international conference on advanced robotics* (Vol. 13, pp. 533-538).

Zhu, H., Liu, Y., Zhao, J., & Wang, H. (2015). Calculating the medial axis of a CAD model by multi-CPU based parallel computation. *Advances in Engineering Software*, *85*, 96-107.

Zilberstein, S. (1996). Using anytime algorithms in intelligent systems. AI magazine, 17(3), 73-73.

Zucker, M., Kuffner, J. J., & Branicky, M. S. (2007, April). Multipartite RRTs for Rapid Replanning in Dynamic Environments. in Proc. IEEE International Conference on Robotics and Automation (ICRA), 2007