University of South Carolina
Scholar Commons

Theses and Dissertations

Spring 2022

On Providing Efficient Real-Time Solutions to Motion Planning Problems of High Complexity

Marios Xanthidis

Follow this and additional works at: https://scholarcommons.sc.edu/etd

 Part of the Computer Sciences Commons, and the Engineering Commons

Recommended Citation

Xanthidis, M.(2022). *On Providing Efficient Real-Time Solutions to Motion Planning Problems of High Complexity.* (Doctoral dissertation). Retrieved from https://scholarcommons.sc.edu/etd/6864

This Open Access Dissertation is brought to you by Scholar Commons. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact digres@mailbox.sc.edu.

On Providing Efficient Real-Time Solutions to Motion Planning Problems of High Complexity

by

Marios Xanthidis

Bachelor of Science
National and Kapodistrian University of Athens, 2015

Submitted in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy in

Computer Science and Engineering

College of Engineering and Computing

University of South Carolina

2022

Accepted by:

Ioannis Rekleitis, Major Professor

Jason M. O'Kane, Major Professor

Marco Valtorta, Committee Member

John R. Rose, Committee Member

Nikolaos Vitzilaios, Committee Member

Tracey L. Weldon, Interim Vice Provost and Dean of the Graduate School

© Copyright by Marios Xanthidis, 2022
All Rights Reserved.

ACKNOWLEDGMENTS

This dissertation, as every other dissertation ever submitted, is a collective effort of many that interacted and assisted the author in many ways. I would like to thank my parents, my advisors, relatives, and many friends and colleagues for supporting me in ways well beyond their responsibilities. Especially, I would like to thank the US taxpayers, who through the National Science Foundation funded my research and allowed me to work on what I love.

ABSTRACT

The holy grail of robotics is producing robotic systems capable of efficiently executing all the tasks that are hard, or even impossible, for humans. Humans, undoubtedly, from both a hardware and software perspective, are extremely complex systems capable of executing many complicated tasks. Thus, the complexity of many state-of-the-art robotic systems is also expected to progressively increase, with the goal to match or even surpass human abilities. Recent developments have emphasized mostly hardware, providing highly complex robots with exceptional capabilities. On the other hand, they have illustrated that one important bottleneck of realizing such systems as a common reality is real-time motion planning.

This thesis aims to assist the development of complex robotic systems from a computational perspective. The primary focus is developing novel methodologies to address real-time motion planning that enables the robots to accomplish their goals safely and provide the building blocks for developing robust advanced robot behavior in the future. The proposed methods utilize and enhance state-of-the-art approaches to overcome three different types of complexity:

1. *Motion planning for high-dimensional systems.* RRT$^+$, a new family of general sampling-based planners, was introduced to accelerate solving the motion planning problem for robotic systems with many degrees of freedom by iteratively searching in lower-dimensional subspaces of increasing dimension. RRT$^+$ variants computed solutions orders of magnitude faster compared to state-of-the-art planners. Experiments in simulation of kinematic chains up to 50 degrees of freedom, and the Baxter humanoid robot validate the effectiveness of the proposed technique.

2. *Underwater navigation for robots in cluttered environments*. AquaNav, a real-time navigation pipeline for robots moving efficiently in challenging, unknown, and unstructured environments, was developed for Aqua2, a hexapod swimming robot with complex, yet to be fully discovered, dynamics. AquaNav was tested offline in known maps, and online in unknown maps utilizing vision-based SLAM. Rigorous testing in simulation, in-pool, and open-water trials show the robustness of the method on providing efficient and safe performance, enabling the robot to navigate by avoiding static and dynamic obstacles in open-water settings with turbidity and surge.

3. *Active perception of areas of interest during underwater operation*. AquaVis, an extension of AquaNav, is a real-time navigation technique enabling robots, with arbitrary multi-sensor configurations, to safely reach their target, while at the same time observing multiple areas of interest from a desired proximity. Extensive simulations show safe behavior, and strong potential for improving underwater state estimation, monitoring, tracking, inspection, and mapping of objects of interest in the underwater domain, such as coral reefs, shipwrecks, marine life, and human infrastructure.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

From Talos, the giant mechanical protector of Crete, to robotic systems in modern popular culture, robots with exotic capabilities performing inhuman tasks with ease have been an intriguing concept for thousands of years. In particular, some robotic systems exceed human capability, exercising effortless control over multiple components to complete complex tasks all within challenging environments, and these are simultaneously the most exciting to humanity's imagination and most inspiring to scientists throughout the ages. Popular examples of such systems are shown in Figure 1.1.

For generations upon generations, such systems were only introduced in mythological or artistic contexts. However, novel developments in the field of Robotics and Computer Science in recent decades have allowed the systematic study and the discovery of the foundations for developing and materializing such complex robotic systems.

There are many — very meaningful but generally simple — tasks that could be successfully executed by simple robots [10–14]. For such tasks, such as cleaning the floor, demining, or even waking-up heavy sleepers, simple robots can perform robustly and reliably; introducing more complicated robots would not only be an extravagance but also a less reliable option. On the other hand, although simple robots could exhibit complex behaviors, there are more tasks for which robots could be useful that require robotic systems of high complexity. Especially for tasks for which mobility is crucial, humans are essential due to the efficient and complex mobility of the human body — an overactuated mobile manipulator which has 244 degrees of freedom [15]. However, the capabilities of human operators are also stressed and limited in many real-world scenarios, such as in the

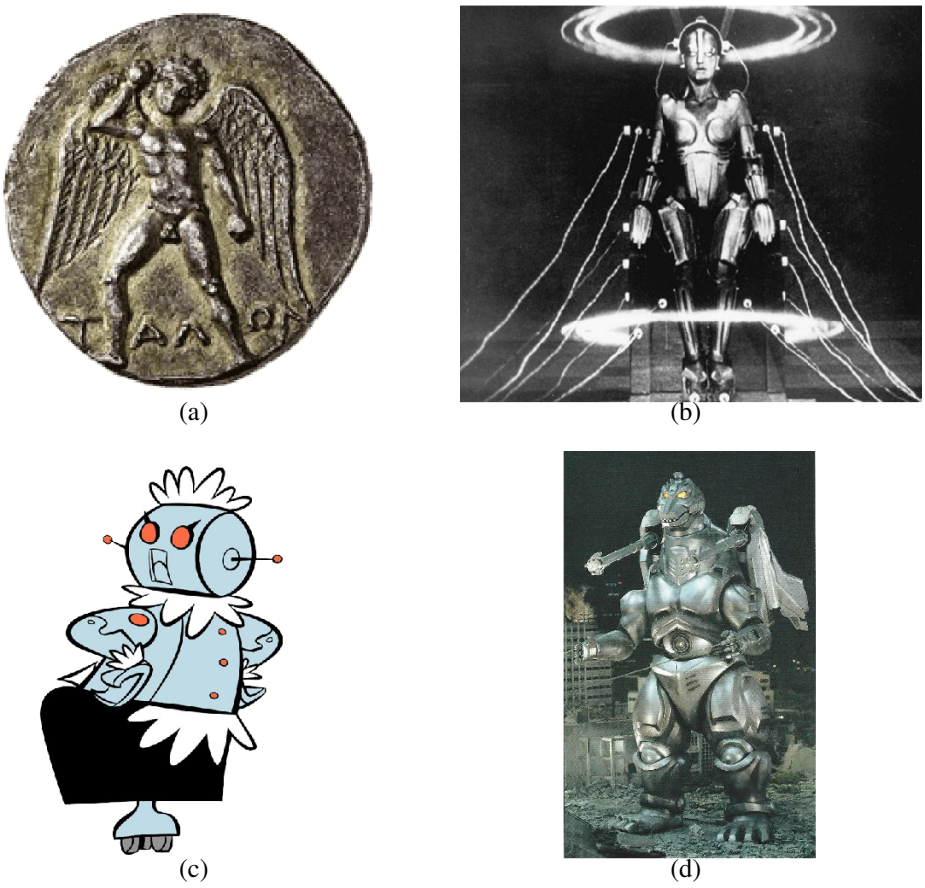
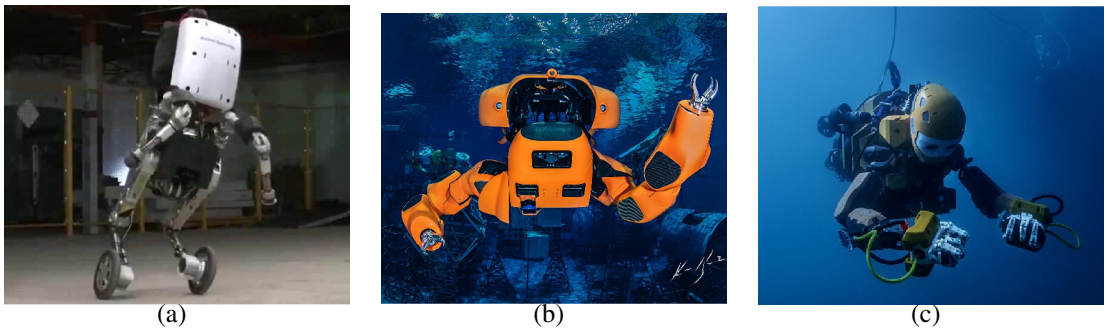


Figure 1.1 Mythological and fictional complex robotic systems from pop-culture: (a) Talos from Greek mythology (Bronze Age), (b) Maria from "Metropolis" (1927), (c) Rosie from "The Jetsons family" (1962), and (d) Mechagodzilla from "Godzilla vs. Mechagodzilla" (1974).

underwater domain, especially when operation in such scenarios is combined with different underlying objectives and multitasking.

Developing complex robotic systems in the present is not only for the sake of excitement or futurism. Rather, it has become essential to the development and sustainability of our society, with simple robots already assisting human workers or users at home or in the production pipeline. Furthermore, the automation of complex, repetitive, hard, or even dangerous tasks for human operators has been a leading idea for many manufacturers and distributors. Some examples of systems following this trend can be seen in Figure 1.2



(a) (b) (c)

Figure 1.2 Real complex robotic systems: (a) Boston Dynamics' Handle, (b) Houston Mechatronics' Aquanaut, and (c) Stanford's Ocean One [1].

Notably, given the hard situations that the SARS-COVID19 pandemic created for our societies by interrupting manufacturing and distribution processes, autonomy has become an even more essential concept that is driven by necessity.

Additionally, a great interest arose in the last decades for operations in the underwater domain, serving various fields and disciplines. Firstly, underwater archaeologists are interested on discovering, documenting and retrieving underwater archaeological artifacts. In the case of shipwrecks, this is a constant battle with time, given the constant deterioration due to underwater conditions. Secondly, marine biologists are focusing on minimal disturbance while studying marine life and the coral reefs' ecosystems, crucial for the health of our planet. It is also very important that essential underwater infrastructure be regularly inspected and maintained to avoid financial or environmental catastrophes, particularly within the energy sector or in aquaculture. Finally, in the defence and security sector, the vastly unexplored underwater domain has been a primary focus for decades in order to guarantee civilian or operational safety on multiple fronts, such as deminining, search and rescue, and border surveillance.

At the present time, maritime operations require human operators due to the challenges arising from the underwater domain. But even though the kinematic abilities of the human body are enough to navigate and perform even complex operations, there exist many challenges that limit the extend of operations and risks to the health of the operators. Operations

with human divers are limited to a maximum depth of 600 meters and require essential life-support equipment that significantly reduces the kinematic and sensing capabilities of the operators progressively as a function of the target depth. Additionally, operations during deep dives are very time-limited at the target depth with respect to the overall dive duration. Last but not least, small hardware failures, miscalculations, bad weather, underwater currents, wildlife or even terrains with cluttered obstacles (such as shipwrecks or underwater caves) constantly place at great risk not only the missions but also the operators. These are challenges that a capable complex autonomous system will be able to overcome, and in the case of a failure, the system can avoid the expense of human lives.

In general, from a computational perspective, for any operation that could be automated, a typical robotic system is developed by addressing and successfully solving two fundamental problems:

- State Estimation, which is the problem of producing estimates regarding the state of the robot and environment, and
- Motion Planning, which is the problem of producing a sequence of motions for the robot to achieve a specific goal, assuming State Estimation is solved to a satisfactory level.

Robotic systems that are capable of operating successfully in challenging environments need to be able to solve both problems in real time so that the system will be able to react robustly, intelligently, and rapidly to the changes of the environment. For robots of high complexity, the motion planning problem is considered their computational bottleneck, with even state-of-the-art methods failing to perform satisfactorily in real-time.

The contribution of this thesis is a set of novel enhancements and frameworks that provide real-time solutions to the motion planning problem for systems of high complexity. More specifically, the contributions include three methods that solve three different facets of this problem:

1. RRT$^+$ [16]: A family of novel sampling-based techniques that plan rapidly, in real-time, for robotic systems with many Degrees of Freedom (DoFs). They were tested against modern state-of-the-art planners in simulation for a kinematic chain and the Baxter humanoid robot, solving challenging motion planning problems orders of magnitude faster.

2. AquaNav [17]: A novel light-weight real-time replanning framework that combines state-of-the-art sampling-based techniques and path optimization methods for efficient offline and online 3D autonomous underwater navigation. It was applied on an agile Autonomous Underwater Vehicle (AUV) with no complete dynamics model called Aqua2 [18], and it was tested on challenging scenarios in simulation, in-pool, and open-water trials. To the author's knowledge, AquaNav is the first framework for underwater robots that enables 3D navigation in complex unstructured terrain with strong safety guarantees that take into account the shape of the robot, and real-time replanning offered by a robust but computationally light pipeline.

3. AquaVis [19]: An extension of AquaNav that employs, for the first time, an AUV with an arbitrary multi-sensor configuration to navigate safely while also observing multiple points of interests, extracted automatically, from a desired distance.

Although not explicitly presented in the thesis, all the above contributions were informed, motivated, and inspired by other relevant works of the author on vision-based state estimation [20–23], localization [9, 24], reconstruction [25], and kinematics [26].

The thesis is structured as follows: Chapter 2 discusses the background on motion planning, RRT$^+$ is described in Chapter 3, AquaNav in Chapter 4, and information about AquaVis is shown in Chapter 5. Finally, the thesis concludes with a discussion in Chapter 6 and conclusions in Chapter 7.

CHAPTER 2

BACKGROUND

2.1 THE MOTION PLANNING PROBLEM

Motion planning is the fundamental problem of enabling autonomy for a robotic system to decide a sequence of actions in order to transition safely from an initial state to a target state or to a target collection of states (region) that accomplish a task, by avoiding the obstacles of the environment. It is also referred in the field as the piano movers problem, Figure 2.1.

Typically, the motion planning problem is formulated as a state space search problem [27], where the state corresponds to instances of relevant properties of the robot, and the state space is formed by the collection of all possible states. In practice, the state of the robot is represented by a vector consisting of all the relevant variables of the system. In the context of robotics, each such independent variable that contributes to the mobility of the robot and describes a single simple kinematic ability is called a *degree of freedom* or DoF. For example, DoFs could be variables that describe revolute or prismatic joints for manipulators, or world frame coordinates and velocities for mechanical systems such as cars, Figure 2.2. DoFs can be controllable, meaning that a control command could directly and independently affect it, or passive, meaning that they are affected indirectly by controlling other DoFs. Given the state space and a query, the result of a successful search is a sequence of valid states that, if followed by the robot, the goal region is reached.

The state space is formed based on the DoFs, since by definition, each DoF contributes one variable, thus adding one dimension to the problem. Then, every point of the state space represents a state of a robot, and it is often called a *configuration*. The set of all pos-

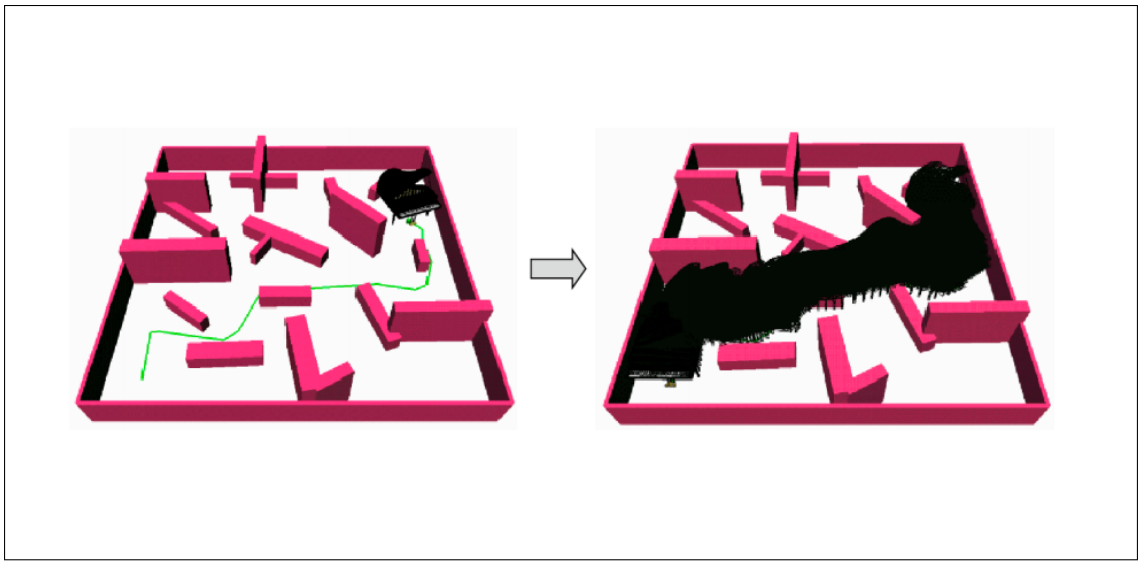


Figure 2.1 A solution for an instance of the piano movers problem. The goal is to produce the motions which should be applied to the piano to move it from an initial to a goal position. Figure was borrowed from Kuffner and LaValle [2].

sible such states of a robotic system is called the *configuration space*, commonly denoted as $\mathscr{C}$. Examples of configuration spaces for different systems are shown in Figure 2.3.

The configuration space could be discrete, continuous, or less often a collection of volumes. If the configuration space is discrete, then it is simply a set of distinct states. An example of a discrete continuous space could be a robot bounded to move on a 2D grid, with a discrete transition model, thus the robot can only get into specific finite positions. If the Configuration Space is continuous, meaning that all the DoFs considered are continuous variables, then, as a mathematical concept, it is a space with volume and other geometrical properties, often non-Euclidean. An example of a continuous $\mathscr{C}$ is a robot moving on a surface, with no discretized states were the robot can freely move, thus the robot can be anywhere on the surface. A configuration space is a collection of volumes, if it is formed by both continuous and discrete actuators, introducing discontinuities. A $\mathscr{C}$ that consists of a collection of volumes could be achieved by combining DoFs that form a continuous space and DoFs with discrete values that are forming a discrete space – such systems are called hybrid. An example of such system could be a robot that moves on a 2D surface

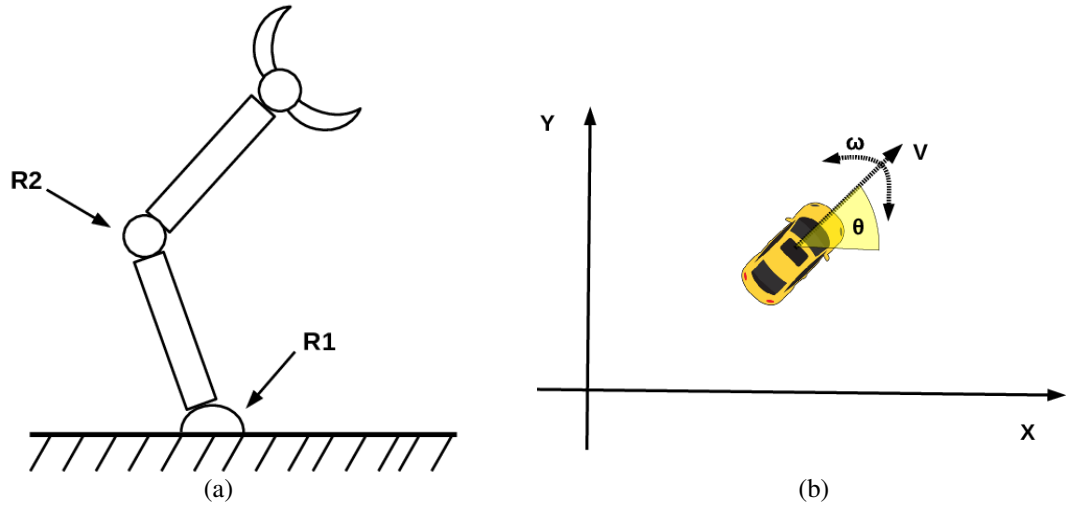


Figure 2.2 The Degrees of Freedom for (a) a 2R manipulator, and (b) a car. The state of the manipulator can be described using the angles of the revolute joints R_1, R_2, and the state of the end-effector if considered, and for the car with the position X and Y, the orientation θ, and, potentially, the angular velocity ω and linear velocity V.

but at the same time carries a module that can move only in discrete positions, such as a gripper, which could be only in an open or closed configuration.

In this thesis, the main focus will lie on motion planning for systems with continuous configuration spaces, and discretizations will be avoided for two important reasons:

- Most real world robotic systems and applications need to perform robustly on real, thus continuous, environments, and often discretization fails to sufficiently model the reality.

- Even when the environment and the system could be modeled discretely to an acceptable resolution, by discretizing the actions, the true capabilities of the robot are often underutilized. This could lead to less efficient solutions, or even to a decrease in the probability that a solution will be found.

Motion planning problems can also be classified not only with respect to the continuity of $\mathscr{C}$ but also their transition model into:

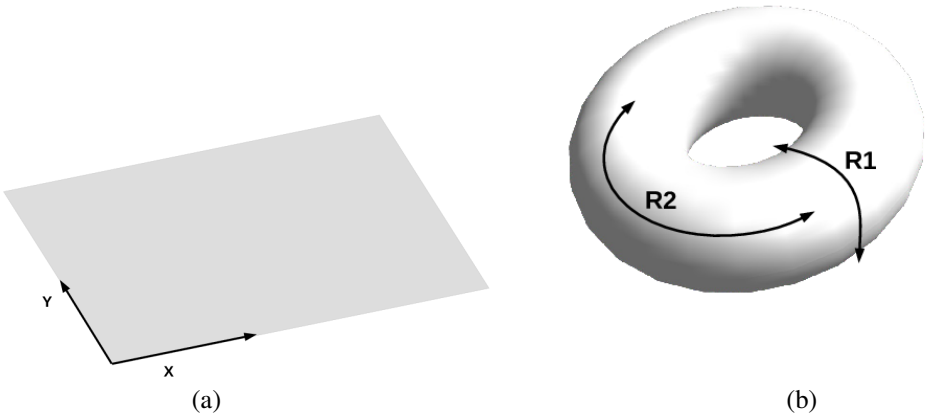


Figure 2.3 Example of two different Configuration Spaces: (a) for a point robot moving in 2D, and (b) for a 2R manipulator, such as the one depicted in Figure 2.2-a, with joints that can perform full rotations. The $\mathscr{C}$ at (a) is formed by simply the 2D coordinates of the robot, thus it results into a Euclidean plane. On the other hand the $\mathscr{C}$ of the manipulator consists of two angles, forming a non-euclidean doughnut-like space.

- Holonomic, where the number of DoFs is equal to the number of the controllable DoFs.
- Non-Holonomic, where the number of DoFs is larger than the number of the controllable DoFs.

That means that if a system is holonomic, every DoF is controllable, and could be controlled directly and independently, so from any given state of the $\mathscr{C}$ the system can potentially transition to other states at any direction. An example of such system could be manipulators where each joint is controlled independently, such as the one shown in Figure 2.2-a. However, if a robotic system is non-holonomic, there exist DoFs which cannot be controlled directly, thus from any given state of $\mathscr{C}$, the system can transition only to a subset of $\mathscr{C}$. An example of a non-holonomic system could be a car, where lateral motion is impossible, such as the one shown in Figure 2.2-b.

Another dimension to classify motion planning problems is whether only kinematic or dynamic constraints are taken into account. Kinematic constraints refer to constraints while transitioning from one state to the other, placed by the limitations of the actuators of the robot, for example due to angular restrictions to joints, or the environment due to obstacles. Dynamic constraints also restrict mobility in real world systems due to other attributes, such as inertia, center of mass displacements, etc. In the first case, only a kinematically valid path needs to be found which the robot could follow without regard to attributes that are affected by motion, where in the second case, the robotic system is subject to velocities, accelerations and external forces that are potentially limiting the available transitions. An example of such system could be aggressive autonomous driving, where safe transitions require effort to satisfy acceleration and velocity bounds.

For any given motion planning problem where obstacles are present, $\mathscr{C}$ consists of two complementary subsets:

- $\mathscr{C}_{\text{free}}$ that consists of all the valid states where the robot is not in collision with the obstacles of given environment or in other invalid states, and
- $\mathscr{C}_{\text{obs}}$ which consists of all the invalid states of the system due to collisions or other constraints.

Even for simple problems, mapping $\mathscr{C}$ directly from the workspace could become counter-intuitive and extremely challenging, as shown in Figure 2.4.

The motion planning problem could be formed as a search problem with the goal of finding a path between two configurations that lie entirely in $\mathscr{C}_{\text{free}}$, Figure 2.4. The hardness of the general motion planning problem is well-known, proven to be PSPACE-complete [28]. The runtime of even the best known exact search algorithm is exponential in the dimension of the configuration space [29].

Thankfully, advances in the field of Robotics and Computer Science have led to different approaches that robustly solve difficult and challenging motion planning problems.

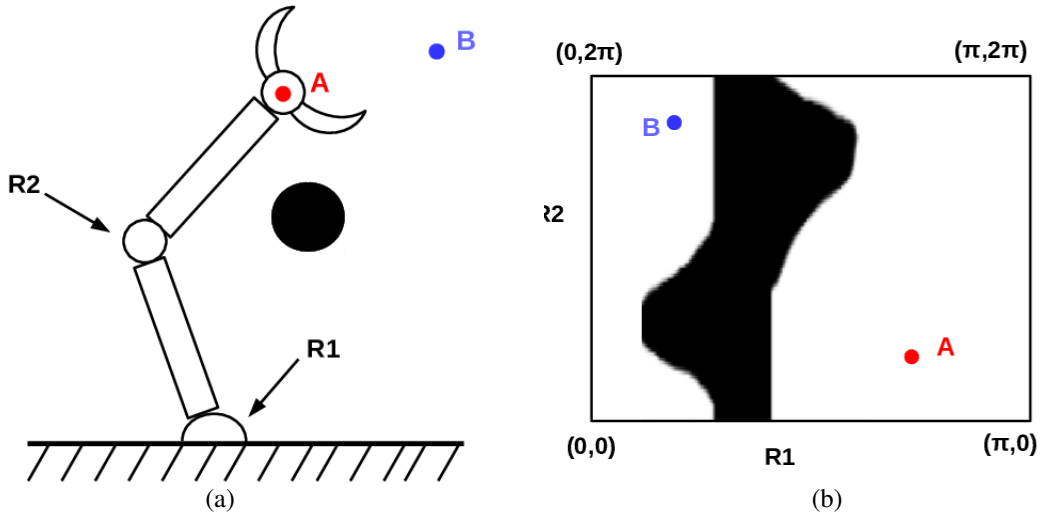


Figure 2.4 An example instance of (a) the planning problem an (b) the unfolded approximate $\mathscr{C}$-space for a 2R manipulator, with R_1 being able to rotate only from 0 to π. At (a) the black circle indicates and obstacle that produces the $\mathscr{C}_{\text{obs}}$. The manipulator starting at configuration A should achieve configuration B, but there is no possible path within $\mathscr{C}_{\text{free}}$.

Most approaches can be categorized within three big families, which are discussed in detail in the next sections.

2.1.1 Sampling-Based Planners

Sampling-Based planners are randomized methods that aim to find a path between an initial and goal configuration by picking samples, mostly at random, to explore $\mathscr{C}_{\text{free}}$ and use them to form graphs. If the initial and the goal configurations are nodes of such a graph, and they both belong in the same connected subgraph, then the motion planning query can be reduced to a simple path search on the subgraph between these two corresponding nodes. Major representatives of planners in this category are PRMs [30] and RRTs [31].

If in the resulting graph, the initial and goal nodes are not in a common connected subgraph, then it is inconclusive whether a solution, exists or not. Thus, sampling-based planners are not complete methods in strict terms, but they might have Probabilistic Completeness. Probabilistic Completeness means that assuming a solution for a planning query

exists, the possibility of discovering it tends to 1 as the execution time of these methods tends to infinity. The guarantee of the Probabilistic Completeness is proven for both PRMs [30] and RRTs [31] with uniform sampling.

At first glance, Probabilistic Completeness might seem a weak guarantee by giving the impression that some problems might require searching for an indefinite amount of time and, unless a solution is found, we can never decide whether a solution exists or not. Indeed, in theory these are two issues that sampling-based planners fall short for some problems, but in practice they are used with a timeout, and they are capable of finding solutions for many planning problems sufficiently fast. When they finish by reaching a sufficient timeout without discovering a solution, then we can safely assume that a solution is unlikely to exist, or unlikely to find easily. Due to the random nature of these methods, restarting them could have beneficial results [32].

Sampling-based planners can be categorized with respect to the reutilization of the graph they are constructing and the optimality of the solution they provide. The next sections will address the multiple and single query planners along with their variances that guarantee optimal or near-optimal solutions.

MULTIPLE QUERY PLANNERS

Sampling-based multiple query planners focus on producing a data structure tailored to a specific environment that can be used to plan for multiple queries producing solutions from different initial to goal configurations [30, 33–35].

The major representative of such planners, that has inspired many other variants, are the Probabilistic RoadMaps or PRMs [30]. The original PRM planner operates in two phases: The learning and the query phase, Figure 2.5.

During the learning phase, a number of random samples are picked from $\mathscr{C}$ with some timeout. Then, the samples in $\mathscr{C}_{\text{obs}}$ are discarded and the ones in $\mathscr{C}_{\text{free}}$ are utilized to form a graph by connecting neighboring nodes with each other within a predefined proximity.

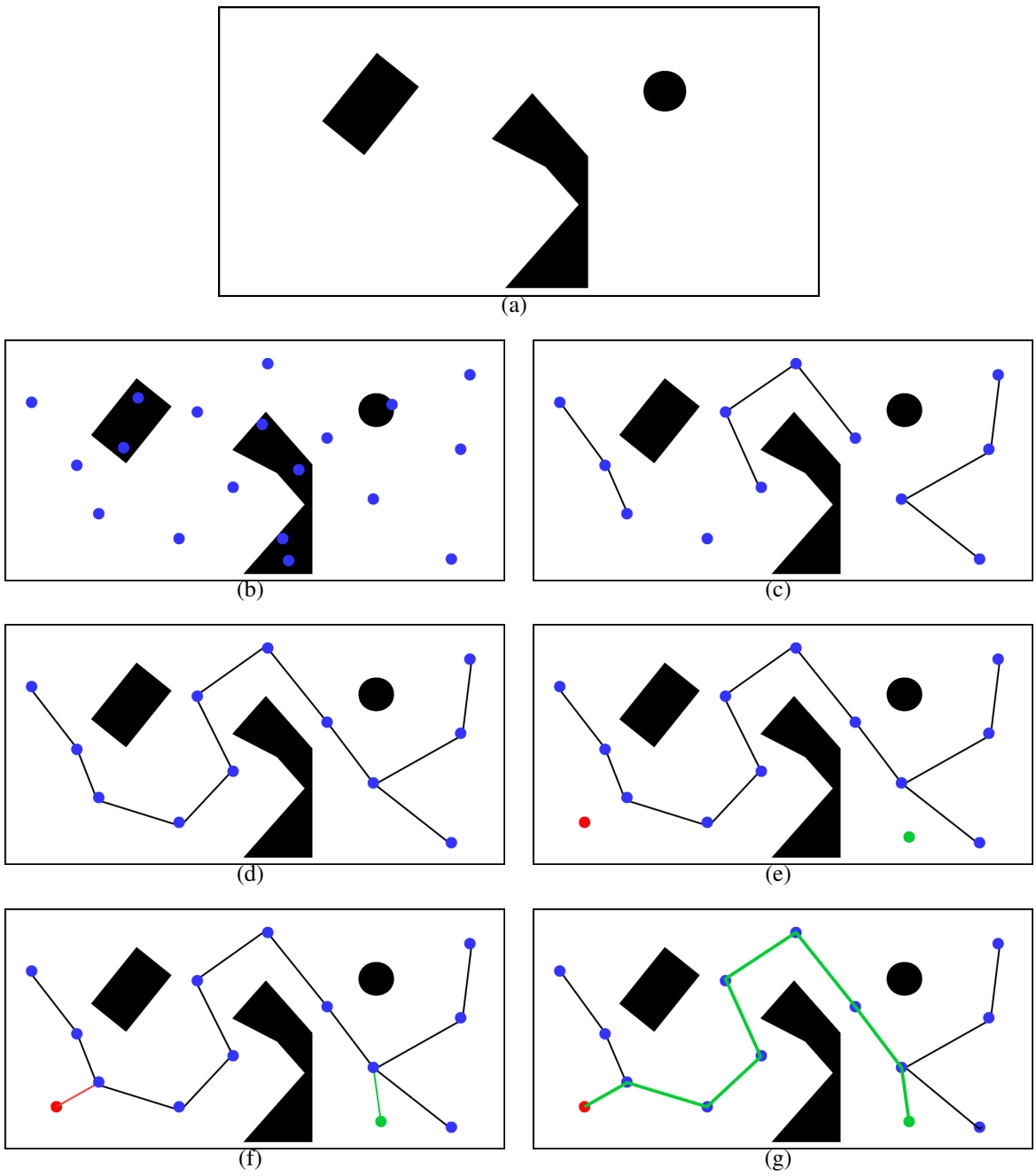


Figure 2.5 An environment example shown at (a), the PRM learning process at (b-d), and PRM query process at (e-g). Initially in the learning process, random samples are drown from $\mathscr{C}$ (b), nearby nodes in $\mathscr{C}_{\text{free}}$ are connected (c), and finally a connected graph in $\mathscr{C}_{\text{free}}$ is produced (d). During the query process, the initial (red) and goal (green) configurations are added (e), connected to the graph with the closest nodes (f), and finally a path solving the planning problem is found (g).

The edges are added in the graph if and only if the transitions, provided by a local planner, lie entirely in $\mathscr{C}_{\text{free}}$. In order to determine the collision-free samples and transitions, a collision checker is needed, typically as a boolean function classifying configurations or path segments as valid or in-collision. In case the result after these operations is a set of disconnected disconnected subgraphs, additional sampling could be deployed to connect these subgraphs and all small disconnected subgraphs are discarded. Lastly, the graph is kept minimal for efficiency by discarding edges and often forming a spanning tree. Reducing the connectivity is important because it affects the run-time performance of path finding queries in the later steps.

During the query phase for given initial and goal configurations, the corresponding nodes are created and connected to the closer valid node of the PRM graph constructed during the learning phase. Then, graph search techniques, such as Dijkstra's algorithm [36], are deployed to find a path connecting the two nodes. When found, the path is returned as the solution to the motion planning query, moving the robotic system from the initial to the goal configuration. The same process is repeated for any other future planning queries.

SINGLE QUERY PLANNERS

Sampling-based single query planners are focused on producing a data structure tailored to a specific query. Such planners have to restart for any future queries. Such techniques focus on building trees instead of connected graphs [37–39]. Expansive Spaces Trees (ESTs) [39] and Rapidly Exploring Random Trees (RRTs) [31] are the major representatives for such methods, with RRTs prevailing in the robotics community for the last 2 decades.

The main idea that both ESTs and RRTs follow is establishing the initial configuration as the root of the tree and then expanding the tree in $\mathscr{C}_{\text{free}}$ until a solution is found. The main key difference is that the ESTs "push" the tree by choosing existing nodes of the tree, and expanding them into unexplored regions, while the RRTs "pull" the tree towards random samples selected uniformly in $\mathscr{C}_{\text{free}}$. In general RRTs are faster, easier to implement for

many different problems, and are widely used. For all the above reasons, this section will focus only on RRTs.

As mentioned above, RRTs attempt to expand a tree in $\mathscr{C}_{\text{free}}$ in hopes that the tree will reach towards and connect with the goal. Figure 2.6 shows an example of the RRT process. The tree is initialized with the initial configuration as the first node. Then for each iteration, a new sample is picked randomly from $\mathscr{C}$. The closest node of the tree to this random sample is selected. Starting from that node, a new node is created in the direction of the random node for a predefined incremental distance or until an state in $\mathscr{C}_{\text{obs}}$ is detected.

Similarly to PRMs, RRTs employ a boolean function for collision checking, that returns whether a selected state is in $\mathscr{C}_{\text{free}}$ or $\mathscr{C}_{\text{obs}}$. The process continues iteratively until a solution is found, by successfully adding a vertex in the goal region as a leaf or a timeout has reached without finding a solution. An additional bias towards the goal is often introduced to ensure that the goal will be reached fast, by directly sampling the goal with some probability. When the search is successful, the solution is extracted by traversing the tree from the goal node to the root.

DIFFERENCES BETWEEN PRMS AND RRTS

As discussed in the previous sections, although both PRMs and RRTs are probabilistically complete techniques that solve the problem of motion planning, PRMs focus on processing multiple queries utilizing the same graph, while RRTs focus on processing a single query by creating a tree for each one. This fact has a number of very important consequences regarding the strengths and the shortcomings of each method, which are not easily noticed at first glance. Indeed many variants of both methods have been introduced mitigating many shortcomings, but most of them introduce new assumptions with different trade-offs, so the discussion will focus strictly on comparing the general algorithms as presented above.

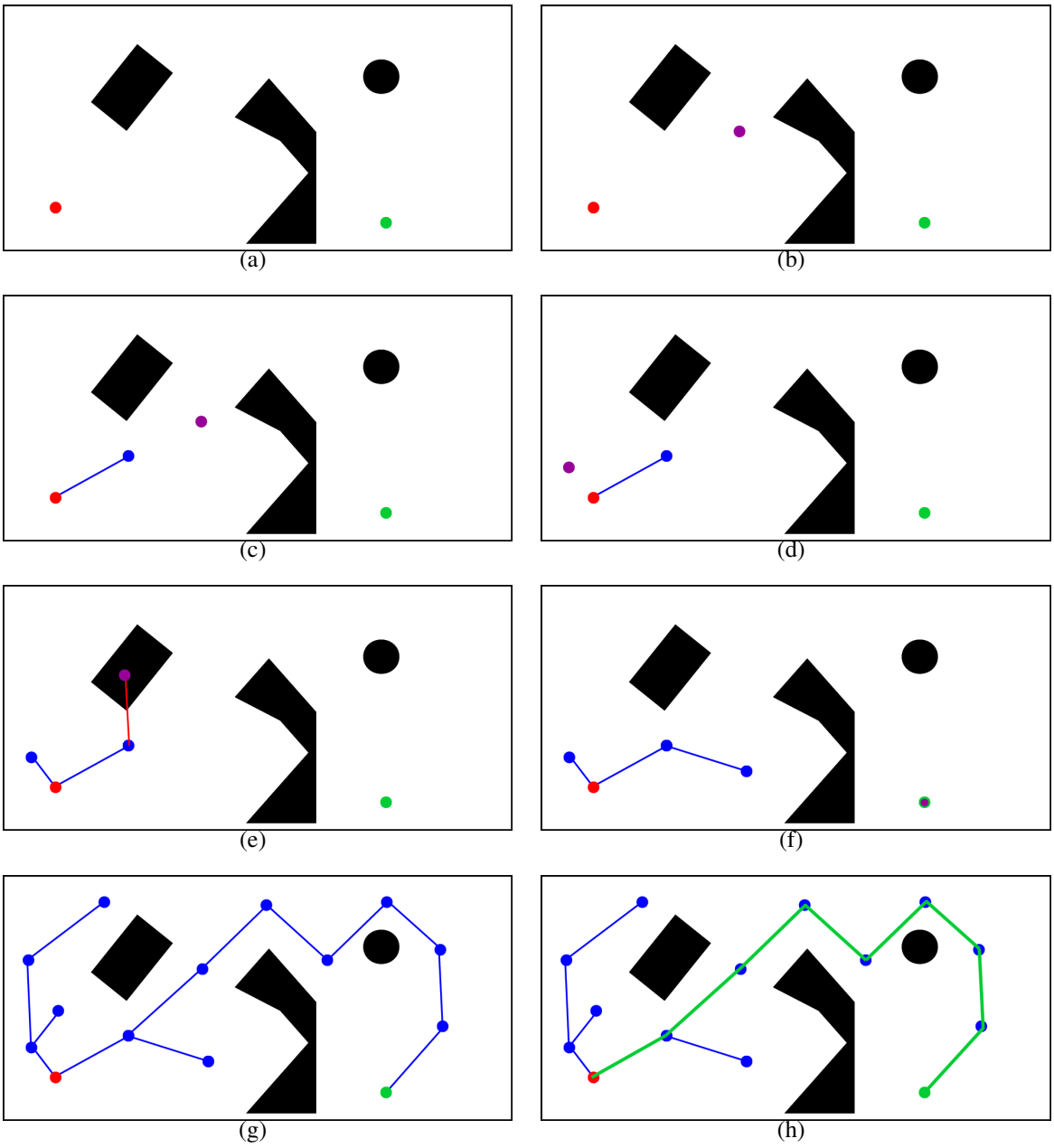


Figure 2.6 An example of the RRT execution for a given query. (a) shows the query with the obstacles indicated with black, and the red and green circles indicating the initial and goal configuration respectively. At (b) a random sample is picked shown with purple, the tree is extended towards it at (c), and similarly for (d). At (e) the tree cannot extend to that random sample due to detected collision. The goal is samples with a bias in order to drive the search efficiently at (f). The search is completed at (g) and the solution is returned by traversing backwards the tree from the goal at (h).

To start with the most noticable one, PRMs are ideal for a well mapped static environment were a robot moves freely undisturbed. They can find solutions for multiple consecutive goals much faster by utilizing the same graph, while for the same problems RRTs will need to restart building a tree from the beginning for each different query. But PRMs are often impractical for cases where the environment is subject to changes, which is a very common feature of real world scenarios. The reason is that PRMs are based on obtaining a valid graph on the learning phase using collision checks. Even the smallest change in the environment can result on nodes of the graph and edges that were initially considered in $\mathscr{C}_{\text{free}}$ change to $\mathscr{C}_{\text{obs}}$, altering the output of the collision checker, thus the solution produced in the query phase could be invalid. On the other hand, RRTs, by restarting each time for different queries, are naturally adapted to the current state of the environment, so assuming the changes in the environment are tracked, RRTs will always provide valid solutions.

Another point of difference is that in general, PRMs require more time to provide a solution to a motion planning query, while RRTs are in general faster. There are two main reasons why this is happening: Number of samples needed and operations needed for each sample.

On the first front, PRMs sample uniformly, seeking to add enough nodes to sufficiently represent the whole of $\mathscr{C}_{\text{free}}$. Thus, the number of samples is expected to be higher. Also, potentially many subgraphs are constructed in regions that might not contribute to the solution, while they might even be in unreachable subspaces of the $\mathscr{C}$ with respect to the initial and goal configurations. RRTs, though, can provide solutions very fast, even while using only few samples that aggressively extend the tree with Voronoi bias, while at the same time, no computational effort is spent on unreachable areas of the $\mathscr{C}$.

Regarding the operations needed for each sample, for each iteration, RRTs need a search to find the closest node to the randomly sampled configuration, which can be executed with linear complexity, and can be significantly improved to logarithmic using K-D Trees [40]. Then the algorithm extends to a new node by checking whether the new edge

is in $\mathscr{C}_{\text{free}}$ or by extending greedily to the furthest possible valid node. Checking the transitions for collisions or finding the furthest valid nodes, for some problems, could be constant time operations for each obstacle, especially when known and well-studied geometric polytopes are used.

In the general case, especially when useful and efficient workspace decompositions cannot be found, the collision checking is performed by consecutively checking single states produced by interpolating (often linearly) from the closest node to the random sample. Of course, in such cases the collision checking method has resolution completeness and the complexity of evaluating an edge is typically linear in the length of the edge.

For PRMs, in order to construct the graph for a given sample, more operations are needed since a search needs to find all the nodes in proximity to the random sample, instead of just one. Thus, the simplest implementation has linear complexity similarly to the RRTs, and $O(k+\log n)$ for the k closest nodes and n nodes in total, which requires more computations than RRTs [41]. Since RRTs for each iteration evaluate only one edge, while PRMs k edges, generally PRMs in total are more costly to construct for each sample. Additionally, PRMs perform extra operations and efforts for connecting disjoint subgraphs, increasing the computational expenses further.

The last and most important difference between RRTs and PRMs, is not quantitative and performance related, but strictly qualitative: PRMs, unlike RRTs,are naturally incapable of planning for non-holonomic systems or for systems under kinodynamic constraints.

The graph created by PRMs is an undirected graph where transitions are allowed from and to both nodes of an edge. On the other hand, RRTs produce a directed graph in a form of a tree, where the transitions are strictly performed from the parent nodes to the children nodes. Thus, RRTs can naturally capture transitions that are valid only from one direction, by using a corresponding collision checker, which are so crucial during planning for non-holonomic systems. Similarly kinodynamic constraints can be enforced during collision

checking or extension by picking valid transitions as determined by the kinematics and dynamics of the robotic system.

For all the above reasons, although PRMs are still widely used mostly for operations in known, generally static environments, RRTs are preferred especially for environments that are prone to changes, and for complex systems that with large volumes of $\mathscr{C}$, non-holonomic kinematics, and kinodynamic constraints.

Asymptotically Optimal Planners

The goal of PRMs and RRTs is to produce a feasible solution to transition a robotic system for an initial configuration to a goal configuration. Both techniques have been very successful on solving many problems but the solutions are often jerky, and almost certainly far from optimal.

The first optimal sampling-based motion planners based on PRMs and RRTs, were introduced by Karaman and Frazzoli [3], and termed PRM* and RRT* respectively. The planners produce paths that attempt to minimize a cost-to-go function with the goal of providing a valid path that also minimizes the cost close to optimality, Figure 2.7.

Both PRM* and RRT* keep the basic characteristics of the original planners and they only introduce an extra value carrying the cost-to-come of the node and an extra operation. Also only one extra distance parameter that defines the neighborhood of a node is needed, with the condition that it is strictly larger than the incremental distance used. That operation aims to relink and change the edges of the graph locally, so that the transitions are updated with respect to the most efficient path. The graph that is constructed and iteratively updated is called Rapidly-exploring Random Graph, or RRG for short. Regarding RRT*, the extra updating steps are performed after the new node is added in the tree. An example execution is shown in Figure 2.8.

In the first step, the goal is to find the most efficient way to connect the new node to the tree. For a given update distance metric, all the nodes inside the hypersphere defined by the

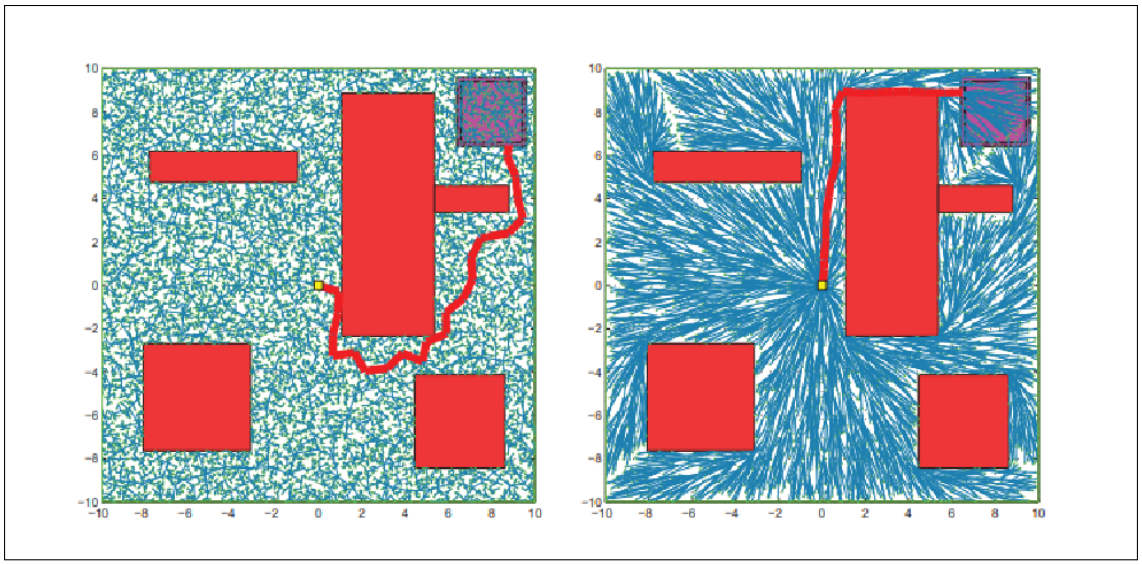


Figure 2.7 A comparison of the RRT (left) vs the RRT* (right) search of the $\mathscr{C}$. As it can be observed, the paths produced by the classic RRT are jerky and far from optimal, while RRT* produces near-optimal smooth paths. Figure is borrowed from the original RRT* paper [3]

update distance and the newly added node are picked, and collision checks are performed for all the edges connecting this node with the ones in proximity. Out of all the edges that are collision free, the one that provides the minimum cost-to-come to the new node is selected and that node is added to the tree replacing the old one, and the cost-to-come is recorded.

The second step is the inverse, and the goal is to find whether by adding this node, more efficient paths are created for the ones in proximity. To achieve that, all the edges starting from the new node to all the neighboring ones are evaluated and the ones that are collision free are picked. Then for every neighboring node, if the cost-to-come decreases by adding the new node and edge, then the corresponding edge replaces the old one and the cost-to-come value is updated.

These small modifications are enough to guarantee asymptotic optimality, meaning that as the execution time tends to infinity, the solution tends to the optimal one. Unlike PRMs and RRTs, the goal of the RRT* and PRM* are not to find a path and then terminate, but they find a path and then iteratively optimize it until a user specified timeout.

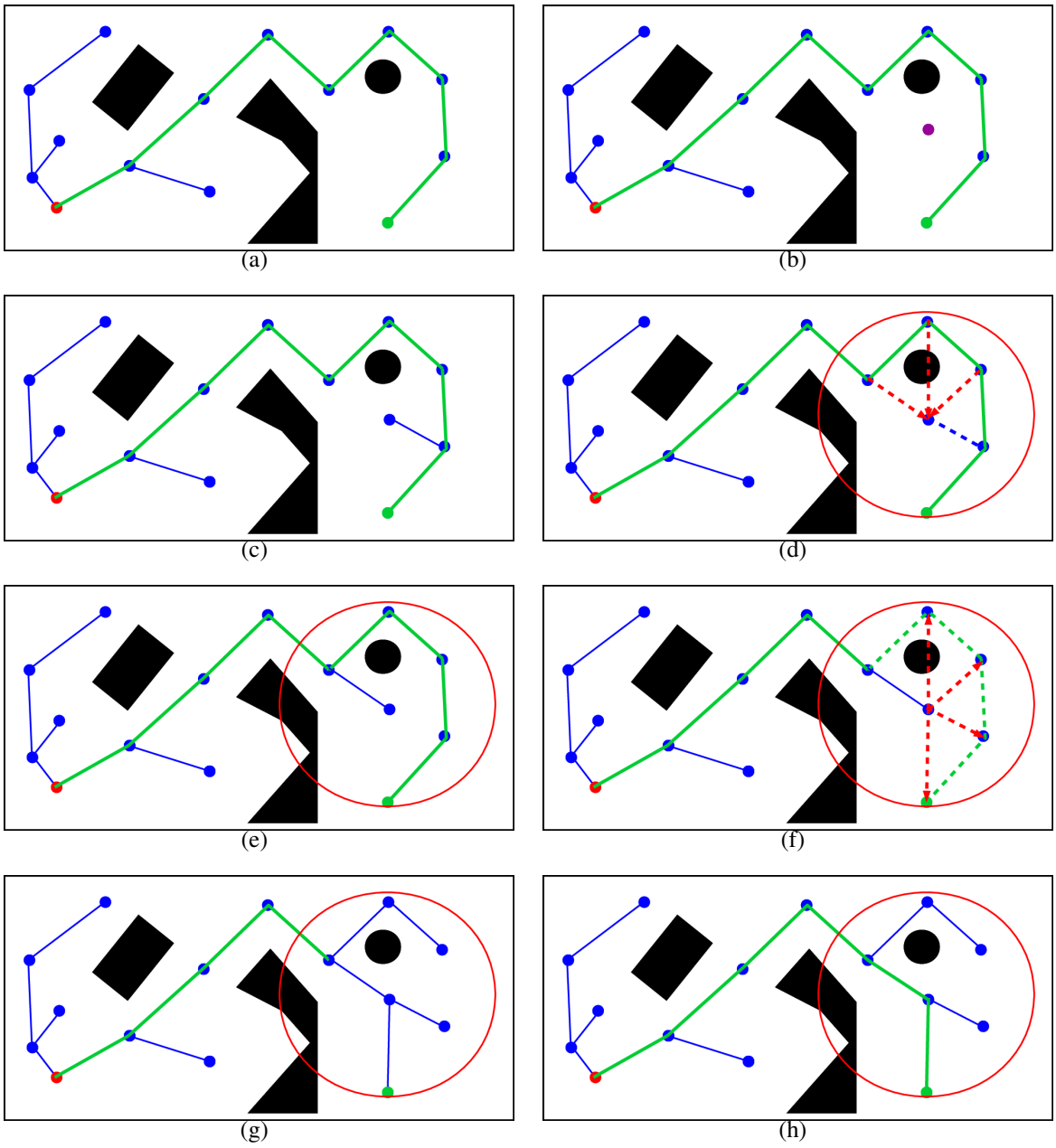


Figure 2.8 An example of the RRT* enhancement. Taking as example the solution of Figure 2.6 (a), a new configuration (purple) is sampled from $\mathscr{C}$ (b). The sample is connected with the closest node (c) and then the RRT* modifications are being applied within a proximity shown with the red circle. Connections to the new node are attempted from the nearby nodes (d) and the one with the minimum cost-to-come is picked (e). Then connections from the new node to all the neighboring ones are attempted (f) and the ones that minimize the cost to go to these nodes are established (g). If the solution is altered the new minimized solution is returned (h),

These approaches are very powerful, since they introduce the notion of any-time motion planning, with the developers being able to decide a timeout, when they can be almost certain that a solution is found and it can be used at anytime, while any extra iterations are only improving the path. But the computational cost for updating and relinking the RRG is in general very high in comparison to the performance of the original planners. Thus applications utilizing such techniques are mostly limited to off-line planning.

However, strong progress in moving towards fast near-optimal solutions has mitigated to a certain extend the negative trade-offs and new asymptotically optimal sampling-based motion planners have been introduced. Some representatives worth mentioning are:

- Informed-RRTs [42] where nodes that will not contribute to a better solution are discarded and the sampling is restricted iteratively into smaller ellipsoids. In general they provide a great accelaration over the RRT* baseline, but they remain generally slow if run long enough due to oversampling.

- Fast Marching Trees (FMT*) [43] that use preselected sets of samples and employ fast graph based techniques to determine the best way to connect them. They are generaly faster, but the solution is resolution dependant, since they operate on preselected sets of configurations. So they need to rerun to provide solutions with better resolution.

- Batch Informed Trees (BIT*) [44–48] that combine both Informed-RRTs and FMT*, by reducing the search space using ellipsoids, while at the same time the work on preselected sets of random samples. They restart the search every time a better solution is found by increasing the resolution of the set. Extensions improve the performance by introducing a specific ordering in the preselected sample set.

2.1.2 Optimization-Based Planners

As shown in the previous sections, sampling-based techniques are probabilistically complete approaches that are able to produce solutions for many problems, but before the introduction of near optimal planners, the paths were highly suboptimal. Execution of the raw solutions with real robots resulted in non-smooth, aggressive and jerky motions, thus efforts were focused on optimizing an existing valid solution to improve the quality of the path with some metric. So, in general, optimization-based planners process as inputs an initial path, specifications for constraints and cost functions, and they output an optimized path given the criteria of the objective function.

Up until CHOMP [4], most path optimization techniques [49–53] required an initial valid trajectory, so a sampling-based planner was always necessary. But CHOMP [4] dropped this requirement for the first time, providing optimized solutions even from invalid initializations. In particular, it was proven very robust for finding solutions for many problems simply by initializing with a linear interpolation between the initial and goal configurations.

In the beginning, CHOMP accepts an initial trajectory as input in a form of sequence of configurations. Then it forms an optimization problem where these variables are the DOFs of each configuration of this sequence. They are subject to constraints on path length and distance from obstacles. The objective function is formed as a combination of cost functions penalizing collisions, keeping kinodynamic constraints during the transitions consistent between the states, and minimizing path length. The problem is solved by using an iterative covariant gradient descent scheme with proper updates to ensure smoothness.

Although CHOMP performs well in many instances, it was shown to be effective mostly for convex problems. Also due the formulation of the cost function, the robots were represented as collections of spheres, with the convex function often requiring more computational resources and for the plans to not allow for high planning accuracy due to that approximation.

Other techniques based on CHOMP have attempted to extend the capabilities of CHOMP [5, 54,55]. Although all these techniques are very capable of solving many problems, successful optimization still relies highly on the initial paths and often local minimum issues are present. Thus, when completeness and guarantees are necessary, path optimization techniques should be used as postprocessing steps to sampling-based planners.

TRAJOPT

Trajopt [5] is a popular path optimization framework that is capable of quickly producing paths that satisfy constraints and optimize desired cost functions. In this section, a short description of Trajopt [5] will be given, since it is one of the core components of AquaNav.

Trajopt models the motion planning problem in a very similar way to CHOMP [4], but it has two major differences (Figure 2.9):

First, instead of solving the optimization problem with projected gradient descent, Trajopt utilizes sequential quadratic programming. There is a major trade-off that Trajopt exchanges with this choice. Every iteration is more computationally expensive than CHOMP, but convergence require fewer iterations. Thus, in general, Trajopt solves most problems faster.

The second difference is between the obstacle avoidance cost function used by Trajopt and CHOMP. CHOMP, as mentioned before, models the shape of the robot as a collection of spheres, thus reducing accuracy. On the other hand, Trajopt operates using the exact shape of the robot, as represented by a triangular mesh, and compares minimum distances to avoid collisions between 3D meshes. Additionally, on that front, Trajopt is less myopic than CHOMP since, instead of trying to simply avoid the obstacles, initially it treats avoiding the obstacles aggressively as hard constraints, making it more capable of avoiding local minimum issues. In more details, comparing precise 3D shapes rather than unions of spheres could be more effective because a single direction to avoid collision is produced for

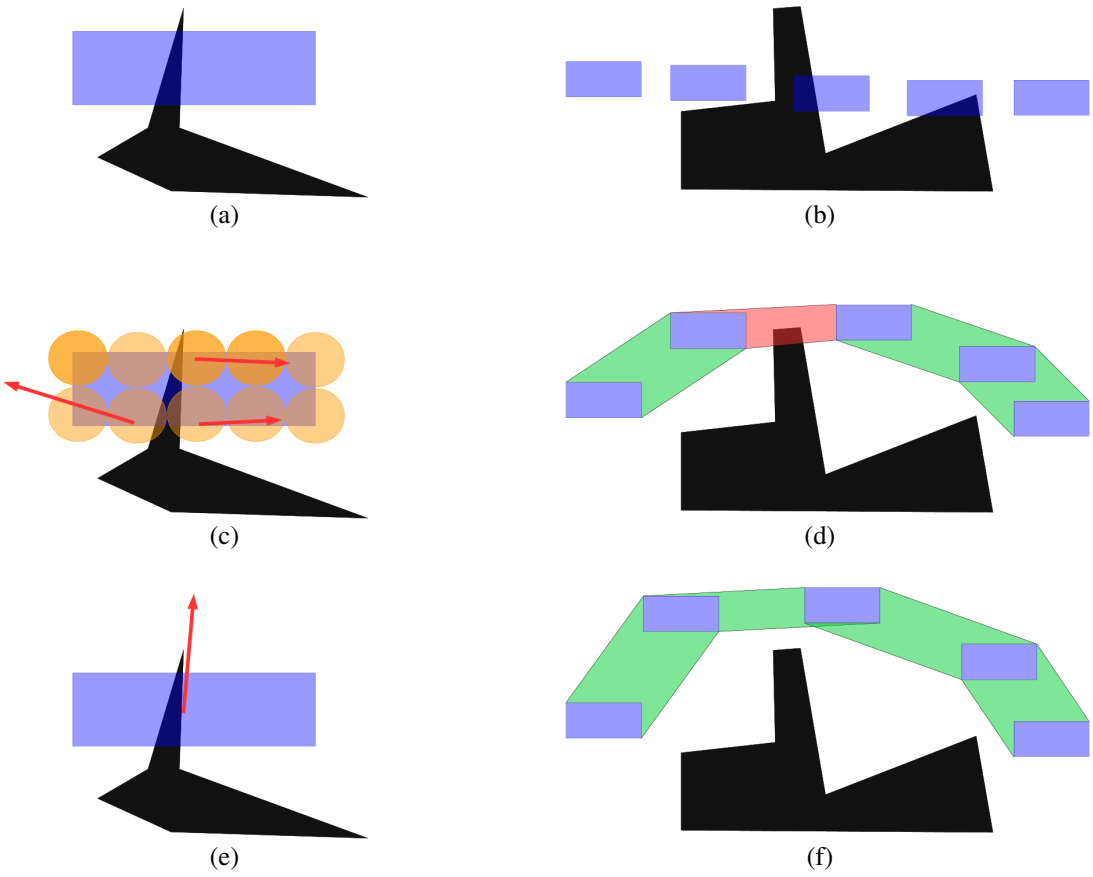


Figure 2.9 Two different examples on the left and the right column respectively, highlighting the differences between CHOMP [4], and Trajopt [5] for local state and global trajectory optimization. The robot is shown with blue, and the obstacles with black. (a-b) show the initial problems, (c-d) the response of CHOMP, and (e-f) the response of Trajopt. For the first one (a) the robot is inside an obstacle. CHOMP's formulation with spheres can produce contradictory costs making optimization more difficult , while trajopt at (e) produces a single cost with the locally optimal direction to avoid collision. The red arrows are indicating the direction of the gradient. For the second one (b), the trajectory passes through obstacles. CHOMP could guarantee that the states will be collision free, but cannot do the same for some transitions (red vs green). But Trajopt guarantees also collision free transitions (f).

every shape in Trajopt, but multiple different directions can be produced for every different sphere for CHOMP.

To conclude, the obstacle avoidance cost function used by Trajopt allows for continuous-time safety by taking into account the swept-out volumes of the robot in the workspace for

every step, thus accelerating convergence and reducing the number of states needed to represent the problem in contrast with CHOMP. This is achieved by appending the obstacle avoidance cost function by using instead of only the mesh of the robot, the convex hull formed by the meshes of each two consecutive states. This formulation ensures that also the in-between transitions of the trajectory guarantee a clearance, which is an important characteristic we employ in the later sections on robot motions prone to oscillations.

2.1.3 SEARCH-BASED PLANNERS

Search-based Planners, also called lattice-based planners, or planning motion primitives, are methods that reduce the motion planning problem into a graph-search problem by discretizing the action space of the robot.

These methods might not guarantee that they can find a solution (if one exists), such as sampling-based planners, or produce motions that optimize an objective, such as the path-optimization methods. But they can have resolution complete guarantees and, in general, they enable the robots to find sufficient solutions much faster than the other alternatives. Especially, they can be practically very capable for systems under kinodynamic constraints with well known models that need to decide on a trajectory in a matter of milliseconds.

The key idea of such techniques when applied in continuous spaces is to use a finite subset of actions, for which the must has to choose for each step. These actions are applied for a small duration and the exact outcome can be calculated very quickly deterministically for both the forward kinematics or also dynamics. When the resulting motions are pre-computed, they are called motion primitives, and they can be used directly by simply superimposing these motions on specific states. An example execution of such planner can be seen in Figure 2.10.

Starting from a known initial state, with a goal to achieve, search techniques such as A* are used to generate new states by interpolating from the action set, validating them with fast collision checking, and exploring greedily using heuristics. When a path is found,

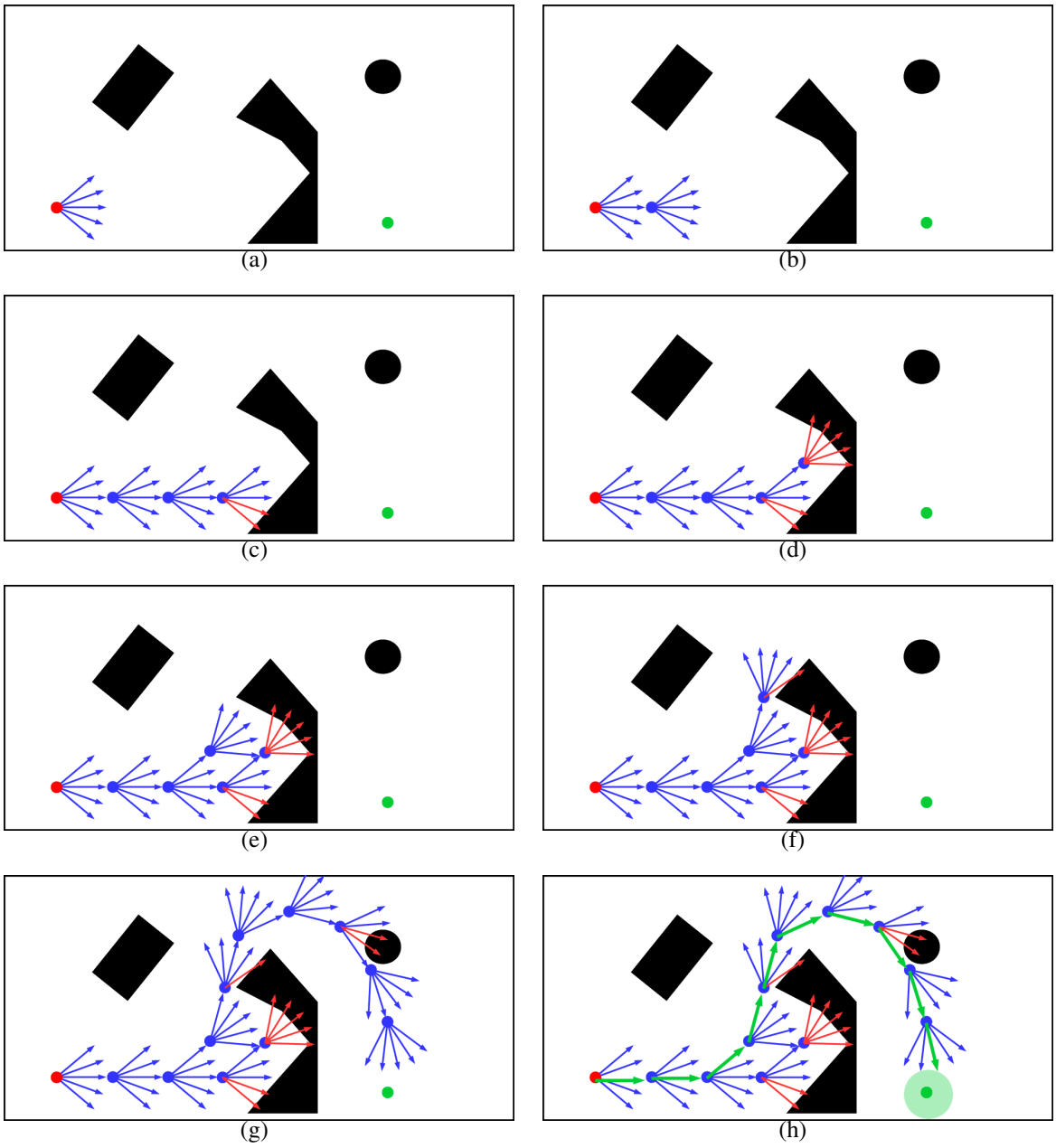


Figure 2.10 An example execution of planning with motion primitives on the problem instance of Figure 2.6-a. Initially, transitions (blue arrows) are applied on the initial configuration (a). Then, greedily the new nodes are explored using a heuristic (b), while transitions that lead to collisions are discarded (c). Then collisions are detected for transitions (d), the search can backtrack and explore alternative paths (e). The process continues (f-g), until a solution (green) reaching the goal to some proximity (light green) is achieved (h).

moving the robot from the initial to the goal configuration, the path is returned as a sequence of actions, and the robot by executing these actions achieves the goal. In motion planning for robots with sensors, new observations are collected and the map is constantly being updated. For such cases variants of A* [56], such as D* [57] and D*-Lite [58] are used, where nodes that were explored before are intelligently updated, saving calculations and improving computational efficiency without degrading optimality guarantees.

Such an approach offers big advantages, by accelerating the planning procedure for even very complicated systems subject to dynamic constraints. Solving a graph search problem is generally orders of magnitude faster than sampling-based and optimization-based techniques. Also, there is no need for complex on-line calculations of kinodynamic constraints, since all the motions considered already satisfy such constraints and their outcome is precomputed offline. Fast and safe response could be achieved easily and only actions that cause collisions should be checked, even for very redundant and high-dimensional systems.

On the other hand, as mentioned in the beginning of the section, such techniques highly underutilize the robot, given that only very few representative motions are considered from the action space. Increasing the number of actions doesn't scale well with real-time performance, thus complex systems might show more suboptimal behavior compared to the other techniques. Given these limitations, there are no guarantees that the motion primitives chosen are capable of solving a problem that could be trivial for the other methods discussed in the section. Also, even the specific goal is not guaranteed to be reached, and more often a goal area is considered, within the goal is considered achieved.

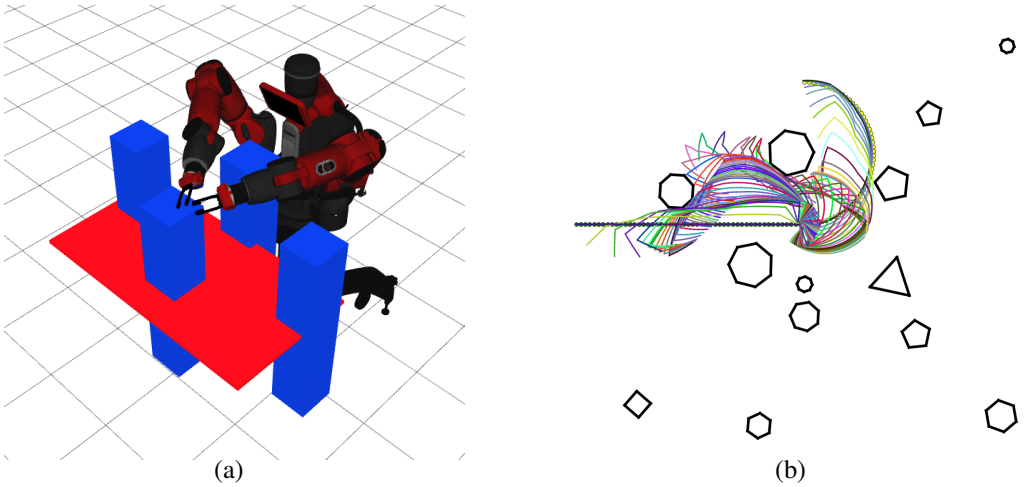
Another issue is that these methods require a well defined heuristic to effectively explore the configuration space, which often is not a trivial problem to resolve. Lastly, search-based techniques can be used only for systems with well-known and well-studied behavior, where the outcomes of the selected actions are captured accurately. This is especially important given that small inaccuracies could lead to error propagation with disastrous results.

CHAPTER 3

RRT$^+$: REAL-TIME MOTION PLANNING FOR HYPER-REDUNDANT SYSTEMS

3.1 INTRODUCTION

Despite the dramatic development of robotics in recent decades, robots are still far from outperforming humans in operations for which they are not specialized. Complex robots such as mobile manipulators, snake robots, and humanoids have been presented mostly in experimental contexts. The fact that the best known mobile manipulator is the adult human body which has 244 degrees of freedom [15], and the fact that modern planners cannot provide plans fast enough for such a system, emphasizes the need for improvement. Sampling-based motion planning algorithms such as rapidly exploring random trees (RRTs) [31] and probabilistic roadmaps (PRMs) [30] are able to avoid explicit reconstruction of the free configuration space, which Canny's algorithm [29] relies upon, but they cannot avoid the underlying curse of dimensionality. Indeed, Esposito's Conditional Density Growth Model (CDGM) for RRTs [59] predicts that the expected number of samples required for an RRT to explore a certain volume fraction with a given probability grows exponentially with the Configuration Space dimensionality. For this reason, even though modern sampling-based planners have exhibited dramatic improvements over the original RRT and PRM techniques, computing real-time solutions for systems with 10 or more degrees of freedom such as hyper-redundant manipulators, snake-like robots, or humanoids, remains a significant challenge. Moreover, this challenge prohibits real-time applications of such high dimensional systems even when only simple kinematic constraints need to be



(a) (b)

Figure 3.1 (a) A Baxter robot in a heavily cluttered environment. (b) A 50-DoF kinematic chain moving amidst obstacles.

satisfied, such as producing trajectories that avoid obstacles. More complex planning problems that require producing paths that satisfy dynamic constraints, or guaranties optimality for some criteria, such as energy efficiency, inertia reduction, etc, are open problems that have yet to be addressed using fast, real-time efficient, methods. But rapidly generating solutions, even if they are highly sub-optimal, would provide great utility for planners such as InformedRRT* [42] and Batch Informed Trees [44] that rely on initial paths to reduce the search space, and to fast path optimization-based planners such as TrajOpt [5]. In general, techniques such as the above require initial valid paths, in order to avoid excessive computational costs for large search spaces, and/or addressing local-minimum issues that are difficult to overcome.

This chapter addresses the path planning problem in high-dimensional configuration spaces for holonomic systems. The main contribution is introducing an algorithmic enhancement that:

- Substantially accelerates the discovery of solutions for systems with many DoFs, up to two orders of magnitude compared to the original state-of-the-art planners.

- Inherits the fundamental properties of the original planners, such as completeness or the local connection strategy.
- Is general enough to be applied to a large variety of holonomic robotic systems.
- Is scalable and linear with no explicit limitations on the number of dimensions.
- Requires no user-defined mapping functions, although simple generic policies could be developed to provide solutions even faster.

The proposed enhancement, rather than directly enabling planning for systems subject to desired constraints, aims to produce solutions that could be utilized quickly by path optimization methods that require an initial path to produce a desired solution such as CHOMP [4], STOMP [54], or Trajopt [5] variants. More specifically, assuming a feasible initial solution is provided fast enough —which is typically the most time consuming part of the process— path optimization techniques could be deployed to optimize quickly the given path into a desired path that satisfies dynamic or energy efficiency constraints in real-time.

The approach is based on the observation that, for many redundant systems, often only a subset of the kinematic abilities are needed to complete a task [26]. Therefore, we propose beginning the search in a *lower dimensional subspace* of the configuration space $\mathscr{C}$ in the hopes that a simple solution will be found quickly. An important property of these subspaces is that a solution lying entirely inside these subspaces should be feasible, in the absence of obstacles, so the initial and goal configurations must lie inside every subspace. The proposed method, by construction, generates subspaces that satisfy this constraint. After a certain number of samples are generated, if no solution is found, we increase the dimension of the search subspace and continue sampling in the larger subspace. We repeat this process until a solution is found. In the worst case, the search expands to include the full-dimensional configuration space — making the completeness properties identical to the original version of the planer.

To evaluate this approach, we modified three well-established planners — RRT [31], RRT-Connect [2], and Bidirectional T-RRT [60] — to produce RRT$^+$, RRT$^+$-Connect, and Bidirectional T-RRT$^+$, with the $^+$ symbol indicating that the planners are enhanced using the idea described above. All three planners were compared to the original planners and to KPIECE [61] and STRIDE [6]. These planners were tested on a planar hyper-redundant arm, varying from 12 to 30 DoFs, and on a simulated Baxter humanoid robot, both shown in Figure 3.1, utilizing OMPL [62] and the MoveIt! framework [63]. In many cases, our experiments indicate that a solution is found much faster using the proposed approach and the run time appears to be less sensitive to the full dimension of the configuration space. For example, our enhanced version of Bidirectional T-RRT [60] found solutions for the Baxter robot 200 times faster than the original planner, outperforming the other planners we tested by a large margin, providing solutions in real-time. Perhaps surprisingly, the proposed method does not seem to trade-off path length for speed; in most cases, path quality was slightly improved.

The remainder of this chapter is structured as follows: Section 3.2 provides the problem statement and Section 3.3 reviews other approaches for motion planning for high-dimensional systems. Section 3.4 provides technical details of the enhancement, considering important issues such as how the search subspaces are selected, how to generate the samples in those subspaces, and when to expand the search dimension. Section 3.5 presents our experiments applying the enhancement to three well-established planners: RRT [31], RRT-Connect [2], and Bidirectional T-RRT [60].

3.2 Problem Statement

The RRT$^+$ family of planners solves the motion planning problem. In simple words, given an initial state, a goal desired state and environment, the planner should output a sequence of intermediate states that move the robot to the goal region by avoiding the obstacles.

More formally, using the same notation as shown in Chapter 2, let $\mathscr{C}$ denote a configuration space with n degrees of freedom, partitioned into free space $\mathscr{C}_{\text{free}}$ and obstacle space $\mathscr{C}_{\text{obs}}$ with $\mathscr{C} = \mathscr{C}_{\text{free}} \cup \mathscr{C}_{\text{obs}}$. The obstacle space $\mathscr{C}_{\text{obs}}$ is not explicitly represented, but instead can be queried using collision checks on single configurations or short path segments. Given initial and goal configurations $q_{\text{init}}, q_{\text{goal}} \in \mathscr{C}_{\text{free}}$, we would like to find a continuous path within $\mathscr{C}_{\text{free}}$ from q_{init} to q_{goal}.

For purposes of sampling, we assume that each degree of freedom in $\mathscr{C}$ is parameterized as an interval subset of $\mathbb{R}$, so that

$$\mathscr{C} = \left[c_1^{(\text{min})}, c_1^{(\text{max})}\right] \times \cdots \times \left[c_n^{(\text{min})}, c_n^{(\text{max})}\right] \subseteq \mathbb{R}^n. \tag{3.1}$$

Note that we treat $\mathscr{C}$ as Euclidean only in the context of sampling; other operations such as distance calculations and the generation of local path segments utilize identifications on the boundary of $\mathscr{C}$ as appropriate for the topology. The focus of this work is on holonomic systems, thus transitions are allowed from a state A$\in \mathscr{C}$ to another state B$\in \mathscr{C}$ as long as no collisions occur.

3.3 Related Work

The problem of motion planning has been proven to be PSPACE-hard [64]. During the late 1990s, sampling-based methods were introduced and shown to be capable of solving challenging motion planning problems, but without guarantees of finding the solution in finite time [27]. The two most prominent representatives of those algorithms are probabilistic roadmaps (PRMs) by Kavraki *et al.* [30], which are useful for multiple queries in stable environments, and RRTs by LaValle [31], that are more suitable for single query applications. Two other variations of RRTs were used in this study: RRT-Connect by Kuffner and LaValle [2], which extends the tree more aggressively in each iteration, and T-RRT by Jaillet *et al.* [60], which plans efficiently in cost-maps.

Although the performance of these techniques can be affected substantially by the number of degrees of freedom of the system, some studies have used them successfully for a variety of high dimensional robotic systems including hyper-redundant arms, mobile manipulators, multi-robot systems, and humanoid robots.

A method that uses PRMs and finds collision-free paths for hyper-redundant arms was presented by Park *et al.* [65]. Other studies use RRTs for motion planning of redundant manipulators, such as the work of Bertram *et al.* [66], which solves the inverse kinematics in a novel way. Weghe *et al.* [67] apply RRTs to redundant manipulators without the need to solve the inverse kinematics of the system. A study by Qian and Rahmani [68] combines RRTs and inverse kinematics in a hybrid algorithm that drives the expansion of RRTs by a Jacobian pseudo-inverse.

Other works have applied RRTs to mobile manipulators. Vannoy *et al.* [69] propose an efficient and flexible algorithm for operating in dynamic environments. The work of Berenson *et al.* [70] provides an application of their technique to a 10-DoF mobile manipulator.

For multi-robot systems, many sampling-based algorithms have been proposed. The study of van den Berg and Overmars [71] uses a PRM and presents a prioritized technique for motion planning of multiple robots. Other studies use RRT-based algorithms such as the study by Carpin and Pagello [72] which introduced the idea of having multiple parallel RRTs for multi-robot systems. The work of Wagner [73] plans for every robot individually and, if needed, coordinates the motion in higher dimensional spaces. Other studies propose efficient solutions using a single RRT [74, 75].

Sampling-based planning algorithms have been applied to humanoid robots in Kuffner *et al.* [76, 77]. Other studies, such as the work of Liu *et al.* [78], use RRTs for solving the step selection problem for humanoid robots.

Regardless of the application, several studies explicitly attempt to reduce the dependence on dimensionality in sampling-based motion planning. Vernaza and Lee [79] extract

structural symmetries in order to reduce the apparent dimension, providing near-optimal solutions but only for known environments where the cost function is stable. Yoshida [80] tries to sample in ways that exploit the redundancy of a humanoid. Wells and Plaku [81] reduce the dimensionality for 2-D hyper-redundant manipulators by modeling the end-effector as a single mobile robot, and the other links as trailers being pulled. While there are many specialized approaches to reducing the dimension, few of these apply to general robot systems.

Planning in high-dimensional spaces has also been addressed with path optimization techniques such as CHOMP [4], STOMP [54], and Trajopt [5]. These techniques can produce high-quality paths and deal with narrow passages by optimizing an initial trajectory that could be highly infeasible. But often the optimization depends on the initial path and can produce infeasible solutions due to local minima issues. Thus these techniques very often are used as a post-processing step on the result from a time consuming sampling-based motion planner, whose overhead is the focus of our study.

Very recent works propose the application of machine learning techniques to drive the tree growth or produce heuristics so a solution will be found faster. For example, the work of Zha *et al.* [82] proposed a framework based on a Gaussian Mixture Model with reported $30\% - 200\%$ acceleration on the computation speed, in comparison to the original planners. On the other hand, Klamt and Behnke [83] proposed an A* approach with a learned heuristic produced from a Convolutional Neural Network to plan paths efficiently for a high-dimensional robot.

More relevant to our work are planners that attempt to focus sampling in the relevant regions of the configuration space. Gipson *et al.* developed STRIDE [6], an EST-based planner that samples non-uniformly with a bias toward unexplored areas of the configuration space. Yershova *et al.* [84] proposed an approach to focus sampling in the most relevant regions. KPIECE [61] by Şucan and Kavraki uses random 2D and 3D projections to estimate the coverage of Configuration Space $\mathscr{C}$ where the density of samples is lower, provided a

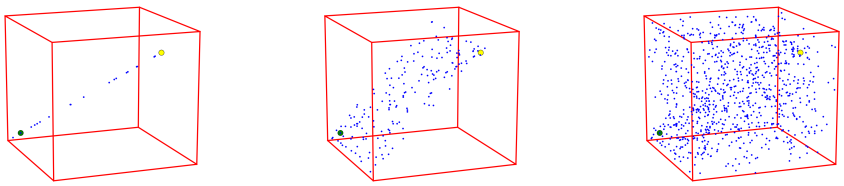
fast planner for high-dimensional configuration spaces. Gochev *et al.* [85] proposed a motion planner that decreases the effective dimensionality by recreating a configuration space with locally adaptive dimensionality. Kim *et al.* [86] present an RRT-based algorithm for articulated robots that reduces the dimensionality of the problem by projecting each sample into subspaces that are defined by a metric. Shkolnik and Tedrake [87] plan for highly redundant manipulators in the low dimensional task space with the use of Jacobian transpose and Voronoi bias. As shown by Şucan and Kavraki [88], even random projections of the configuration space can provide good estimates for its coverage. Recent work of Chamzas *et al.* [89] proposes a novel framework for experience-based sampling, where it decomposes the workspace to local primitives which are stored in a database and on the planning phase corresponding local-planners are synthesized to bias sampling. Lastly, the work of Bayazit *et al.* [90] where a PRM was used to plan in subspaces of the configuration space, creates paths that solve an easier problem than the original, by shrinking the obstacles and then iteratively optimize the solution until the solution becomes valid. Other examples include [6, 71, 73, 79–81, 85, 86].

The observation that high-dimensional systems are often overactuated has been very recently highlighted. Lee *et al.* [91] were able to find efficient solutions by simplifying the kinematic abilities of humanoids, while the method proposed by Jia *et al.* [92] provided a method for solving fast motion planning problems with dynamic constraints by remapping the problem into a grid based search.

This paper introduces the idea of iteratively searching in lower-dimensional subspaces, and emphasizes the potential of using such an approach for efficient motion planning on arbitrary hyper-redundant systems. Unlike previous works, this approach tries to find paths that are not only confined entirely to a subspace but also in a subspace in which a solution can exist, since the initial and goal configurations are part of the subspace. Contrary to the work of Bayazit *et al.* [90], the approach searches in subspaces that are strictly lower-dimensional, leading to faster computation of the solution.

The main advantage of our approach, excluding the major acceleration on the computation of a solution in comparison to the state-of-the-art, is that it can be adapted easily to enhance many of the existing algorithms. The proposed enhancement does not use approaches that do not scale well with the dimensionality, such as grids, and more importantly, no user defined mapping functions are needed, although if applied they could provide even better performance.

3.4 Method Description



(a) (b) (c)

Figure 3.2 Sampling in one, two and three dimensions in $\mathscr{C}$. Red lines indicate the boundaries of $\mathscr{C}$, the yellow dots indicates q_{init}, and the green dots indicate q_{goal}.

3.4.1 Planning in Subspaces

The proposed method is based on searching for a solution in lower dimensional subspaces of $\mathscr{C}$, in expectation that such a path might be found faster than searching in the entirety of $\mathscr{C}$. The underlying idea is to exploit the redundancy of each system for each problem. To achieve this, the algorithm starts searching in the unique linear 1-dimensional subspace of $\mathscr{C}$ that contains q_{init} and q_{goal}. If this search fails, the planner expands its search subspace by one dimension. This process continues iteratively until the planner finds a path, or until it searches in all of $\mathscr{C}$. In each subsearch, the tree structure created in lower dimensions is kept and expanded in subsequent stages.

Algorithm 1: RRT$^+$

Input : A configuration space $\mathscr{C}$, an initial configuration q_{init}, and a goal configuration q_{goal}.
Output: RRT graph G

```
1  C_sub ← 1-d subspace of C, through q_init and q_goal
2  G.init(q_init)
3  while True do
4  |   q_rand ← sample drawn from C_sub
5  |   q_near ←NearestVertex(q_rand, G)
6  |   q_new ←NewConf(q_near, q_rand)
7  |   G.AddVertex(q_new)
8  |   G.AddEdge(q_near, q_new)
9  |   if done searching C_sub then
10 |   |   if dim(C_sub) < dim(C) then
11 |   |   |   Expand C_sub by one dimension.
12 |   |   else
13 |   |   |   return G
```

Algorithm 1 summarizes the general approach as applied to RRT. Lines 2 through 8 encapsulate the typical RRT algorithm [31]. The rest show the proposed modifications, with emphasis on line 4, which calls the novel sampler. These enhancements are implementing the following behavior: The planner starts optimistically by searching in one dimension, along the line passing through q_{init} and q_{goal}. If this search fails to find a path—a certainty, unless there are no obstacles between q_{init} and q_{goal}—the search expands to a planar subspace that includes q_{init} and q_{goal}, then to a 3D flat,[1] and so on until, in the worst case, the algorithm eventually searches all of $\mathscr{C}$; see Figure 3.2.

The description in Algorithm 1 leaves three important elements unspecified. First, the algorithm needs a method for selecting and representing the subspace $\mathscr{C}_{\text{sub}}$ (Lines 1 and 11). Second, a method is required for sampling from this subspace (Line 4). Third, the conditions that must be met before moving to the next subsearch must be defined (Line 9). The choices explored in this study are described in the next sections.

[1] We use the term *flat* to refer to a subset of $\mathbb{R}^n$ congruent to some lower-dimensional Euclidean space.

It is worth noticing that the enhanced planners inherit the transition method of the original planners (Lines 6 and 8), since only the sampling stage is altered. Thus the collision checking and the transition functions are also inherited directly from the original planners, and the proposed enhancement does not affect those elements in any way.

3.4.2 Representing and Sampling from Subspaces

The central idea is to search for solutions in subspaces of progressively higher dimensions. The primary constraint on these subspaces is that they must contain both q_{init} and q_{goal}. Subspaces that violate this constraint cannot, of course, contain a path connecting q_{init} to q_{goal}. In general, the algorithm's selections for $\mathscr{C}_{\text{sub}}$ should ideally be directed by the likelihood that a solution will exist fully within $\mathscr{C}_{\text{sub}}$. However, it is not clear how this likelihood should be computed. Instead, we consider a simple random technique that is quite effective, especially for highly-redundant systems.

The choice that we investigate —one that trades generality for simplicity— is the prioritized release of the degrees of freedom. The idea is that initially all the DoFs will be constrained to vary linearly together, so that the available subspace $\mathscr{C}_{\text{sub}}$ is the single line connecting q_{init} to q_{goal}. Each time the search is ready to expand to a higher dimensional subspace, one DoF is chosen to be released. For the released DoFs, instead of enforcing the above-mentioned linear constraints, they take values from their range, independently from the other DoFs. More formally, given a set $P_{\text{con}} \subseteq \{1,\ldots,n\}$ of DoFs to be constrained, we can form $\mathscr{C}_{\text{sub}}$ by constraining the DoFs in P_{con} to form a line passing from q_{init} and q_{goal} and allowing the remaining DoF to vary freely. In each step, one randomly-selected DoF from P_{con} is removed, thus increasing the dimensionality of $\mathscr{C}_{\text{sub}}$.

Next, the algorithm requires a technique for drawing samples from $\mathscr{C}_{\text{sub}}$. The sampling uses a very efficient linear time method that initially generates a sample within $\mathscr{C}$ along the line between q_{init} and q_{goal} by selecting a random scalar r and applying:

$$q_{\text{rand}}^{(i)} = (q_{\text{goal}}^{(i)} - q_{\text{init}}^{(i)})r + q_{\text{init}}^{(i)}. \tag{3.2}$$

Algorithm 2: Prioritized Sampler

Input : Initial configuration q_{init}, goal configuration q_{goal}, set of constrained DoFs P_{con}, boundaries of the random number r r_{min}, r_{max}.

Output: Sample configuration q

1 $q \leftarrow$ random point in $\mathscr{C}$ along line L from q_{init} to q_{goal} with r_{min} and r_{max}
2 **for** $i \in 1, \ldots, n$ **do**
3 **if** $i \notin P_{con}$ **then**
4 $q[i] \leftarrow \text{Random}(0, 1) * (c_i^{(max)} - c_i^{(min)}) + c_i^{(min)}$
5 **return** q

The algorithm then modifies q_{rand} by inserting, for each DoF not in P_{con}, a different random value within the range for that dimension; see Algorithm 2. More specifically, the sampler starts by having all the DoFs constrained, thus a random sample is generated along line L, and no value is modified in the loop that follows. When a DoF has been removed from P_{con}, then a random sample is selected on the 1-D line L, but the corresponding value of the released DoF is altered with a random value from its entire range. Thus, the sample will be drawn from a 2-D plane extended by the corresponding basis vector of that dimension. The process continues by releasing an additional DoF at every iteration, until a solution is found or all DoFs are released and planning proceeds on the complete configuration space, as with the original planner.

To ensure that the samples along the line L between q_{init} and q_{goal} remain within $\mathscr{C}$, we compute r_{min} and r_{max} using Algorithm 3 and select a scalar r randomly from the interval $[r_{min}, r_{max}]$. The *ComputeBoundaryValues* function calculates the line passing from q_{init} and q_{goal} (lines 1 and 2), finds the intersections of the line with all the different $c_i^{(min)}$ and $c_i^{(max)}$ flats (lines 3 through 5), and returns the limits r_{min} and r_{max} (line 16). When r takes the value r_{min} or r_{max} then the resulting sample is one of the two intersection points of the line and the boundaries of $\mathscr{C}$ (lines 6 through 12). The *ComputeBoundaryValues* function is called only once before the planning loop.

Algorithm 3: ComputeBoundaryValues

Input : Initial configuration q_{init}, goal configuration q_{goal}, dimensionality of configuration space n, limits of configuration space $c^{(min)}$, $c^{(max)}$.

Output: Minimum value of scalar r_{min}, maximum value of scalar r_{max}

```
1  D ← q_goal − q_init
2  L = Dt + q_init
3  for i ← 1 to n do
4  |   for c in {c_i^(min), c_i^(max)} do
5  |   |   Find the intersection p = Dt_p + q_init of L and c
6  |   |   if p ∈ 𝒞 then
7  |   |   |   Find t_p so p = Dt_p + q_init
8  |   |   |   if (t_p ≤ 0) then
9  |   |   |   |   r_min ← t_p
10 |   |   |   else
11 |   |   |   |   r_max ← t_p
12 return r_min, r_max.
```

The prioritized method provides an efficient and easy way to sample from projections of arbitrary dimensionality, while, as stated before, it provides valuable understanding that may be utilized while developing new prioritization policies for each robotic system.

3.4.3 TERMINATING THE SUBSEARCHES

The only remaining detail to be discussed is how long the search in each subspace should continue. A set of timeouts $\{t_1, t_2, \ldots, t_n\}$ is generated in which t_i corresponds to the amount of time spent in the i^{th} iteration. Ideally, the planner should stop searching in subspaces that seem unlikely to provide a solution. For simplicity, in this work the timeouts are precomputed by assuming that the t_i follows a geometric progression. The idea is to exponentially increase the number of samples in each successive subsearch, acknowledging the need for more samples in higher dimensions.

The proposed approach uses two different parameters specified by the user. The first parameter T is the timeout for the entire algorithm. The second parameter $\alpha > 1$ is a factor describing a constant ratio of the runtime between successive subsearches. The total time

Algorithm 4: RRT$^+$

Input : Initial configuration q_{init}, goal configuration q_{goal}, timeout T, factor α.
Output: RRT graph G

$P_{con} \leftarrow \{1, 2, \ldots, n\}$
G.init(q_{init})
r_{min}, $r_{max} \leftarrow$ ComputeBoundaryValues(q_{init}, q_{goal}, n, $c^{(min)}$, $c^{(max)}$)
$\{t_1, t_2, \ldots, t_n\} \leftarrow$ FindSampleSize(T, α, n)
for $i \leftarrow 1, \ldots, n$ **do**
    for t_i time **do**
        $q_{\text{rand}} \leftarrow$PrioritizedSampler($n$, q_{init}, q_{goal}, P_{con}, r_{min}, r_{max})
        $q_{\text{near}} \leftarrow$NearestVertex($q_{\text{rand}}, G$)
        $q_{\text{new}} \leftarrow$NewConf($q_{\text{near}}, q_{\text{rand}}$)
        G.AddVertex(q_{new})
        G.AddEdge(q_{near}, q_{new})
    Extract(P_{con})
return G

T available to the algorithm can be expressed in terms of α and a base time t_0:

$$T = \sum_{i=1}^{n} t_0 \alpha^i. \tag{3.3}$$

Solving for t_0 we obtain:

$$t_0 = \frac{\alpha - 1}{\alpha(\alpha^n - 1)} T. \tag{3.4}$$

Using this t_0 every t_i can be computed as:

$$t_i = \alpha t_{i-1}. \tag{3.5}$$

In this study, good performance was achieved when α took a value between 1 and 2 and T was scaled linearly with the dimensions of the configuration space. Moreover, in the experiments it is shown that the T parameter does not negatively affect the performance beyond a certain value. If the T value is too small, the search in the lower subspaces terminates early and the value of the proposed enhancement is reduced, since the original algorithm's sampling in the entire $\mathscr{C}$ is applied.

Finally, putting everything together, RRT$^+$ is presented in Algorithm 4.

3.5 EVALUATION

For the experiments, three new planners were developed using the OMPL framework [62]. These planners work by applying the proposed technique to three RRT variants: (1) RRT [31] with default goal bias of 0.05, (2) RRT-Connect [2], and (3) the BiT-RRT [60] by assuming a uniform cost-map. The BiT-RRT is intended to test the effect of our method on a powerful cost-map planner.

3.5.1 EXPERIMENTS WITH A 2D HYPER-REDUNDANT MANIPULATOR

In order to test the ability of the new planners to adapt to different problems, the prioritization of the degrees of freedom was chosen randomly for each run. We demonstrate that, given enough redundancy, even a random prioritization provides results much faster (up to two orders of magnitude). A computer with a 6^{th} Generation Intel Core i7-6500U Processor (4MB Cache, up to 3.10 GHz) and 16GB of DDR3L (1600MHz) RAM was used.

OMPL's standard example of a 2D hyper-redundant manipulator, created for STRIDE [6], was used in order to test and compare the enhanced planners in a standard way. Two different environments were tested 100 times each for a kinematic chain with varying degrees of freedom (between 12 to 20): A Cluttered Random environment and a Horn environment; see Figure 3.3. The initial and goal configurations were the same as shown in Figure 3.3, so a qualitative comparison in problems with different redundancies could be done.

In all cases, the enhanced versions of each planner were faster and in most cases significantly faster than the original ones. The average and the median times for the Random Cluttered environment are presented in Figure 3.4, and in Figure 3.5 the same for the Horn environment. As can be seen, as the dimensions of the configuration space increase the proposed enhancement outperforms the original algorithm (lower is better).

The BiT-RRT$^+$ not only outperformed the BiT-RRT by a wide margin but also outperformed all the other planners using uniform sampling. Additionally, it provided competi-

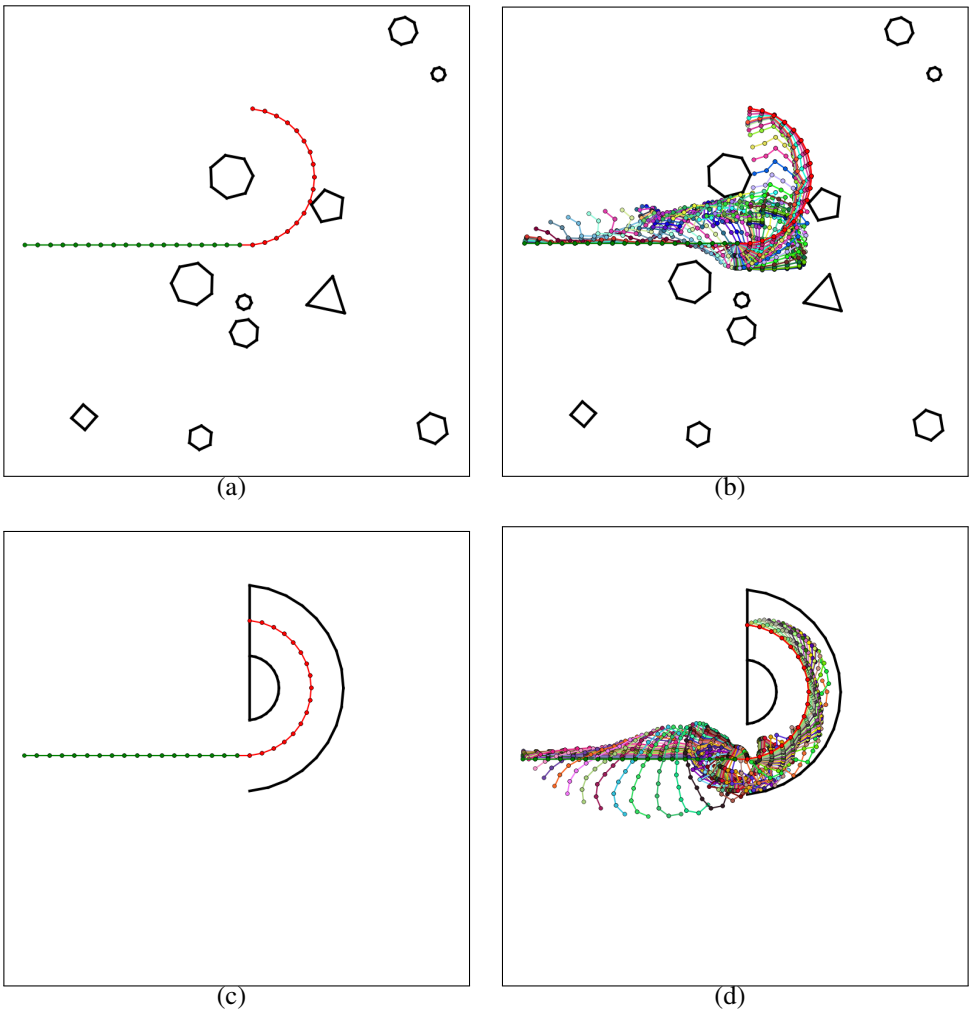
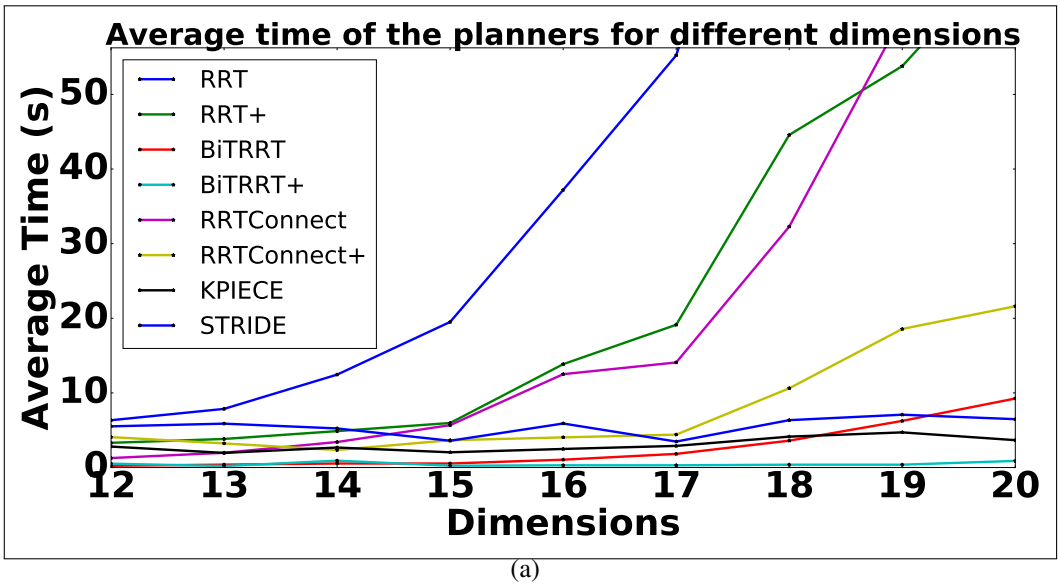


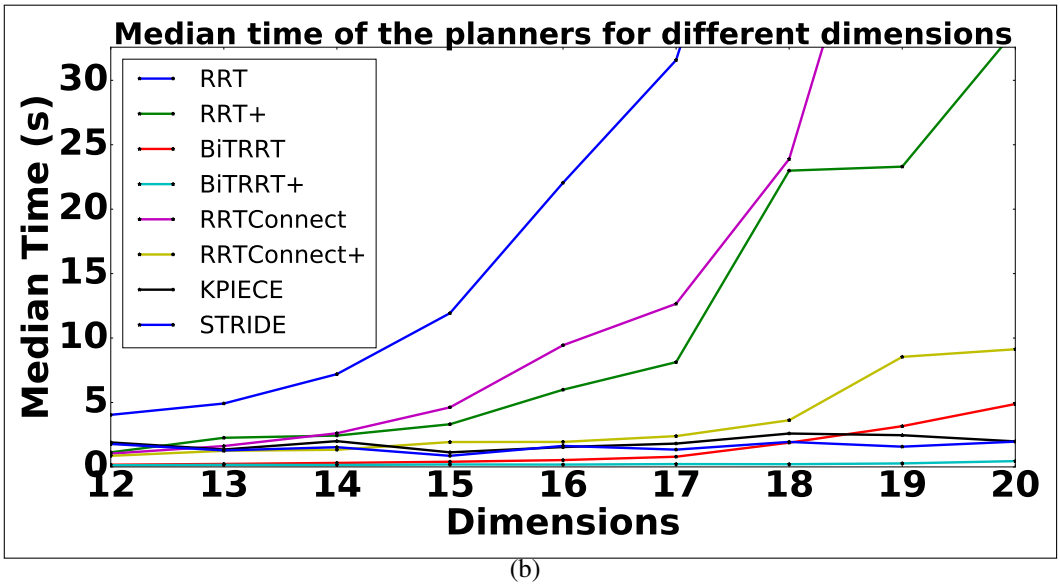
Figure 3.3 The two different environments, called Random Cluttered and Horn [6]. In (a) and (c) the two planning problems are presented. The red chain indicates the initial configuration (q_{init}) and the green chain represents the goal configuration (q_{goal}). In (b) and (d), solutions for the two environments, produced by RRT$^+$-Connect, are shown using a random color palette.

tive results for robust planners with biased sampling such as KPIECE [61] and STRIDE [6], which as expected perform much better in less redundant environments. Interestingly, for each problem, the single fastest solution across all trials was generated by BiT-RRT$^+$. This suggests that a good choice of prioritization may give results faster in a consistent way.

Additional experiments with the fastest planners were performed by varying the number of degrees of freedom between 12 and 30 for the Cluttered environment. The performance of the BiT-RRT$^+$ in Figure 3.6 demonstrated superior performance in increased dimension-



(a)

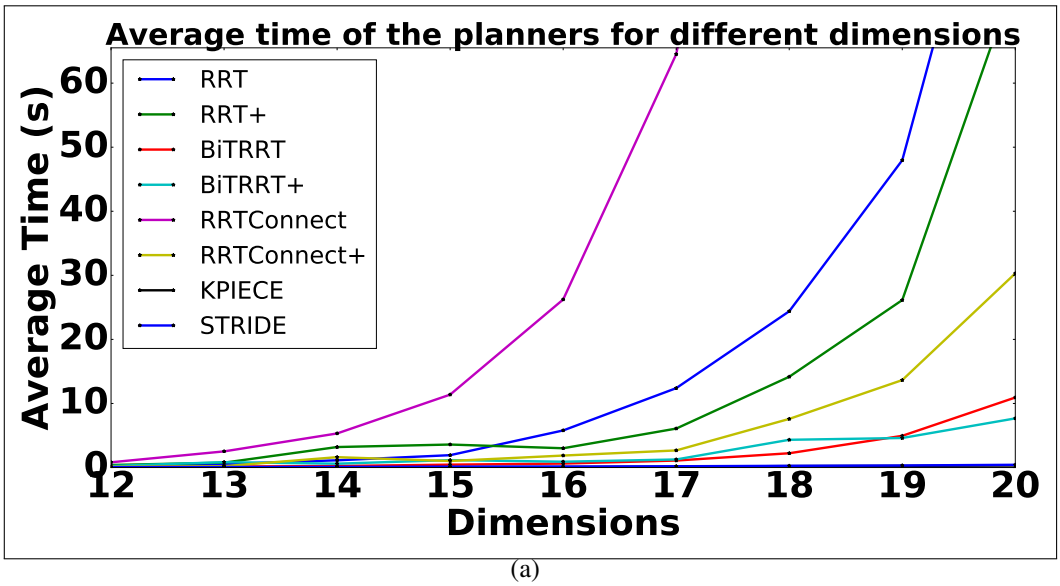


(b)

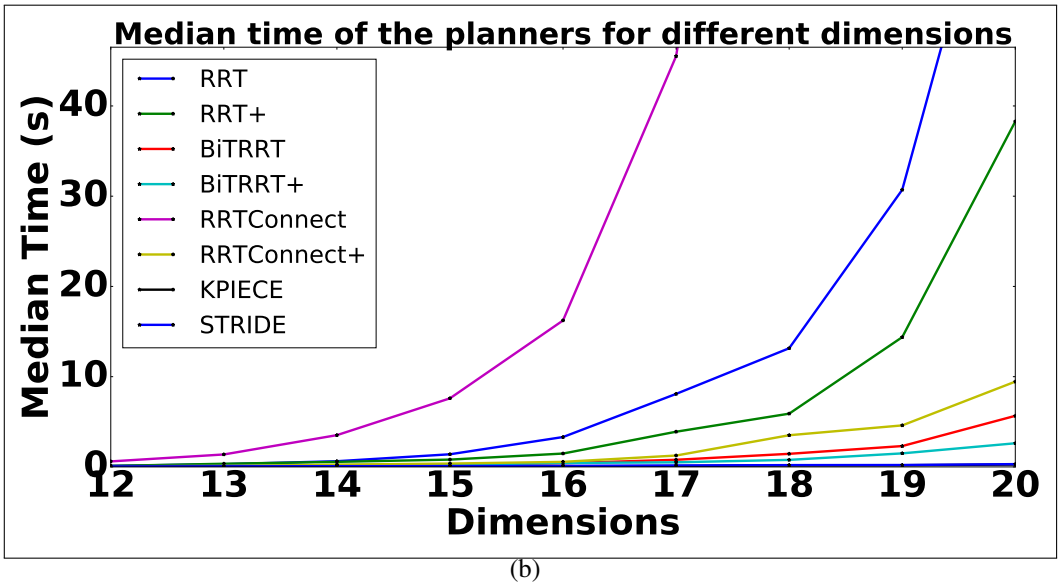
Figure 3.4 The average (a) and the median (b) time for the Random Cluttered environment.

ality. Also the BiT-RRT$^+$ provided the fastest results among all the planners. Even with random prioritization only after the 29 dimensions due to insufficient prioritizations the median run-time of BiT-RRT$^+$ becomes slightly larger than KPIECE and STRIDE.

As shown in Figure 3.7, our method does not significantly affect the path quality as measured by path length.



(a)



(b)

Figure 3.5 The average (a) and the median (b) time for the Horn environment.

We also demonstrate that the performance of the planners is relatively insensitive to T across a wide range, as shown in Figure 3.8. In order to study the sensitivity by including also the outliers, T was used for Equation 3.4, and in case a solution was not found in T time, the planners continued sampling uniformly in $\mathscr{C}$ until a solution was found. Although a good tuning may positively affect the efficiency, no clear trend is observed on the efficiency of the enhanced planner as a function of T.

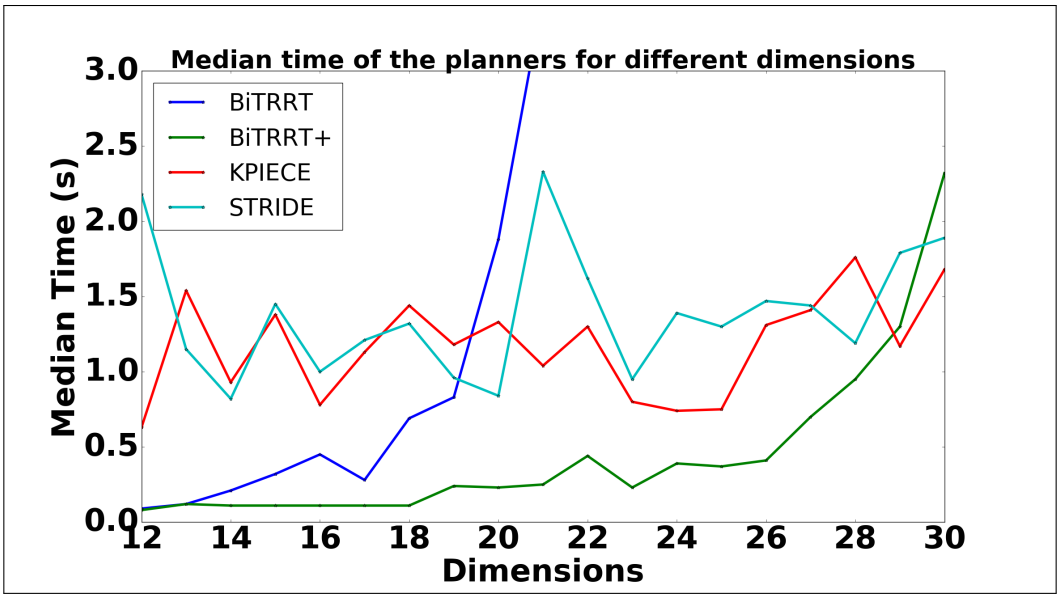


Figure 3.6 Comparison of the median time for the Cluttered Random environment from 12 to 30 dimensions of BiT-RRT and BiT-RRT$^+$, KPIECE and STRIDE. Interestingly, BiT-RRT$^+$ is more than 200 times faster than BiT-RRT for 30 DoF.

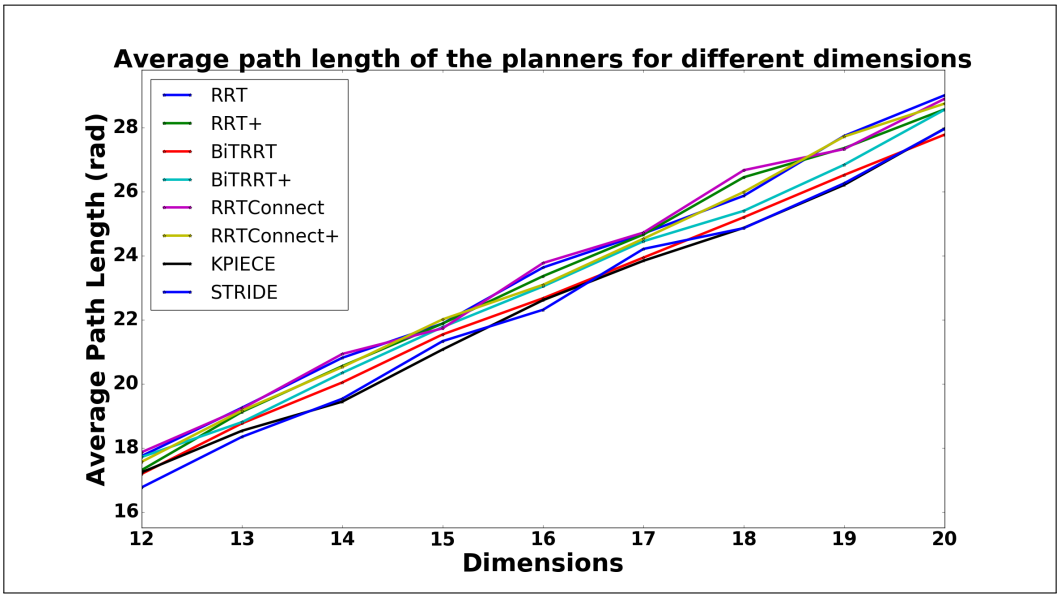


Figure 3.7 The average path length for the Horn environment after the standard path-simplification method of OMPL.

Lastly, we demonstrated further the capability of the enhancement to solve very challenging problems with many dimensions in real-time, by simply using random prioritizations. BiT-RRT$^+$ was benchmarked along with KPIECE [61] and STRIDE [6] on a 50-DoF manipulator in the Cluttered environment (Figure 3.1-b) with a small time-out of only 1

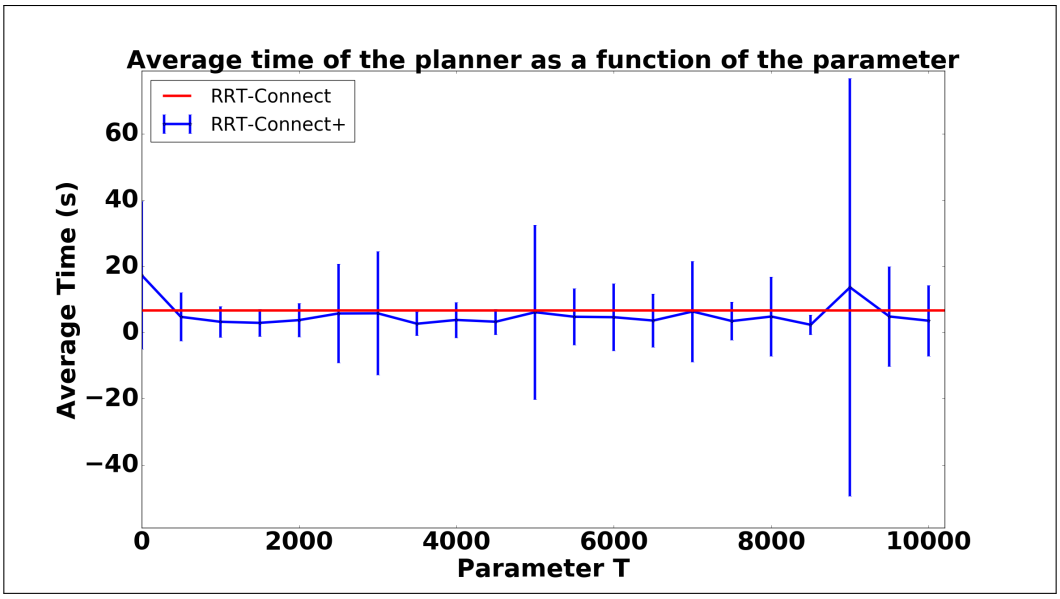
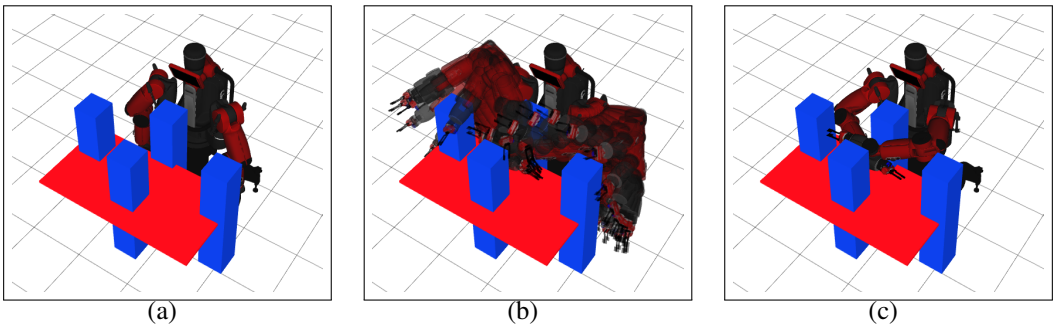


Figure 3.8 Sensitivity to T from 0 to 10000 with step 500, given $\alpha = 1.6$ for 200 runs per value of RRT$^+$-Connect in the Random Cluttered environment for 15 degrees of freedom.

	Success rate (%)	Average (s)	Median (s)
BiT-RRT$^+$	**66**	0.29±0.25	**0.44**
KPIECE	34	0.19±0.26	0.65
STRIDE	39	0.24±0.27	0.58

Table 3.1 Success rate, average and median of 100 trials for the 50-DoF kinematic chain, with a time-out of 1 second. BiT-RRT$^+$ is close to twice more successful than KPIECE [61] and 1.7 times more successful than STRIDE [6], with shorter median computation time for the solution.

second. As shown in the results (Table 3.1) BiT-RRT$^+$ is twice as effective for solving fast problems of such high-dimensionality, with a shorter median time than the other two state-of-art techniques, illustrating the potential of the method on generating paths for real-time applications. Note that the original algorithm, BiT-RRT, is not reported, since it was not able to successfully produce any solution within the given time-out.



(a) (b) (c)

Figure 3.9 The initial (a) and goal (c) configuration, along with the path (b) used in the experiments with the Baxter. The table is indicated with red, and the four pillars are indicated with blue.

3.5.2 Experiments with Baxter

Experiments are presented for the Baxter humanoid robot (Figure 3.1) with 14 degrees of freedom using the OMPL [62] and MoveIt! framework [63] with an Intel i7-7700 8-core processor (3.6GHz), and 32 GiB RAM.

Given knowledge of the system, instead of choosing the prioritizations randomly, a generic task-independent policy was used to show that even naive policies can eliminate the outliers observed in the earlier experiments and lead to superior performance. The policy was giving priority to the joints closer to the base.

We tested the enhanced planners in a cluttered workspace, which consisted by a table and four parallel pillars. As shown in Figure 3.9, Baxter starts with the manipulators in the relaxed position below the table and the goal is to reach the configuration that both arms fit between the pillars. Moreover, in order to make the problem more challenging for the enhanced planners, one of the manipulators should end up above the other, reducing even more the redundancy of the problem and forcing our planners to explore subspaces with high dimensionality. The RRT$^+$, RRT$^+$-Connect, BiT-RRT$^+$ were compared with their original versions, and also with KPIECE and a bidirectional version of KPIECE provided in OMPL, called BiKPIECE, for 100 trials. A timeout of 60 seconds was used for each run.

As shown in Figure 3.10, each enhanced planner outperformed the corresponding non-enhanced planner. In particular, RRT$^+$-Connect and BiTRRT$^+$ provided solutions much faster than BiKPIECE and KPIECE. Non-bidirectional planners had difficulty finding a solution, showing the difficulty of the problem close to the goal region. The enhanced planners, even when they were failing in the case of RRT$^+$, produced solutions very quickly (in less than 5 seconds) showing the advantage of sampling in lower-dimensional subspaces. Similar to the experiments presented in the previous section, the fastest planner among all was the BiTRRT$^+$. Most of the solutions from RRT$^+$- were found in an 8-dimensional subspace, and from RRT$^+$-Connect and BiTRRT$^+$ in a 6-dimensional subspace.

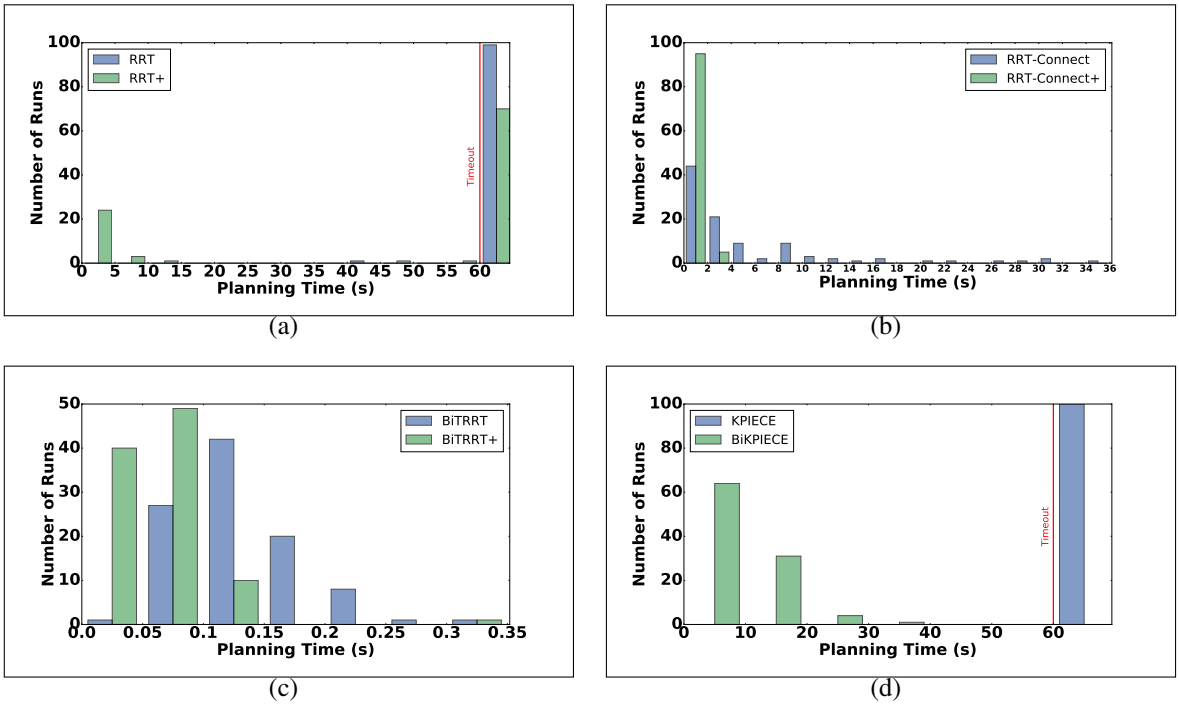


Figure 3.10 The histograms comparing the enhanced planners, with the original ones (a-b-c) along with the histograms of KPIECE and BiKPIECE (d). The red line at 60 seconds indicates the Timeout, and the results after that line should be considered failures. The timeout was chosen to facilitate the large number of tests.

CHAPTER 4

AQUANAV: ROBUST 3D UNDERWATER NAVIGATION FOR AN AUV WITH COMPLEX DYNAMICS

4.1 INTRODUCTION

This section addresses the problem of navigation for underwater structure inspection and mapping. Underwater structure mapping is an important capability applicable to multiple domains: marine archaeology, infrastructure maintenance, resource utilization, security, and environmental monitoring. While the proposed approach is not limited to the underwater domain, this work addresses the challenging conditions encountered underwater, with a special focus on shipwreck mapping. Underwater mapping has traditionally focused on acoustic sensors [93–96]. However, most inspections require visual input [97–99]. The state-of-the-art in autonomous operations is to observe the target structure from far enough to avoid navigation in cluttered spaces, while remotely controlled operations present entanglement hazards. As a result, most inspections suffer from gaps due to occlusions and low-resolution due to the water effects on the camera sensor. The presented work enables autonomous operations underwater close to the structure to be inspected; before, this was only partially possible with teleoperation.

Maps of underwater structures such as wrecks, caves, dams, and docks are often available either through acoustic sensing [100] or via photogrammetry [101, 102]. In this paper we present an augmentation of Trajopt [7], a popular path-optimization open source package for (mobile) manipulators, to facilitate 3D trajectory planning of an AUV utilizing either a known map or an online constructed local map. The proposed method is real-

Figure 4.1 Aqua2 AUV navigating over the Stavronikita shipwreck, Barbados.

ized on an Aqua2 vehicle [18]. The Trajopt planner is augmented with a sampling-based correction scheme that resolves the local minima problem. Furthermore, different map representations were tested including geometric primitives and point-cloud implementations.

AUVs moving in 3D underwater are prone to external forces, such as currents, and inertia. As such, when a robot switches between motion directions, a significant drift may appear. This effect was taken into consideration when designing the safety distance and the linear velocity of the vehicle. When the motions were too aggressive, excessive drift was observed. Utilizing a complete hydrodynamics model of the Aqua2 would resolve such issues, but such a model has yet to be introduced. Previous works [103–105] attempted to provide such analysis but they significantly restricted the mobility of the robot since it was allowed to turn only by no more than one axis each time. The above also makes it infeasible to directly account for the dynamics during planning. It is noteworthy, though, that fast motion planners such as KPIECE [61] that utilize physics simulators to plan with kinodynamic constraints can solve the planning problems for systems with non-trivial dynamics. But considering that the Aqua2 hexapod robot has a very large control space, limited computing resources to accommodate for physics simulators operating in parallel,

and non trivial task-dependant projections —required by KPIECE [61]— the use of such techniques requires excessive resources for a real-time response.

The proposed method was rigorously tested in simulation and in the pool. Utilizing the Gazebo simulator [106] with an underwater extension that emulates the kinematic behavior of the Aqua2 vehicle, the tested environments demonstrated changes in depth and attitude in three dimensions for realizing the produced trajectories. Furthermore, tests at an indoor diving pool, with various obstacle setups, verified the validity of the proposed approach.

The proposed approach provides contributions in two conceptual areas: path planning of mobile robots in 3D and underwater navigation in cluttered environments. More specifically, we augmented the Trajopt package enabling operations of an autonomous mobile robot in three dimensions. We introduced a fast warm-starting method to avoid local minima issues using an RRT-based approach. Furthermore, we demonstrated the use of Trajopt for online planning based on convex decomposition of unordered point clouds. The proposed framework enabled underwater operations in cluttered spaces. In particular, a light-weight geometric navigation framework utilizing the full 3D motion capabilities of the Aqua2 AUV enabled operations with a known map or in unknown areas of cluttered underwater environments. Autonomous operations of real-time planning and replanning were paired with visual inertial state estimation in environments of substantial complexity, even without utilizing a known hydro-dynamics model.

The next section provides an overview of related work. Section 4.2 describes the Aqua2 AUV employed and the problem statement. The proposed approach is detailed in Section 4.4, while experimental results are presented in Section 4.5. The paper concludes with lessons learned and directions of future work.

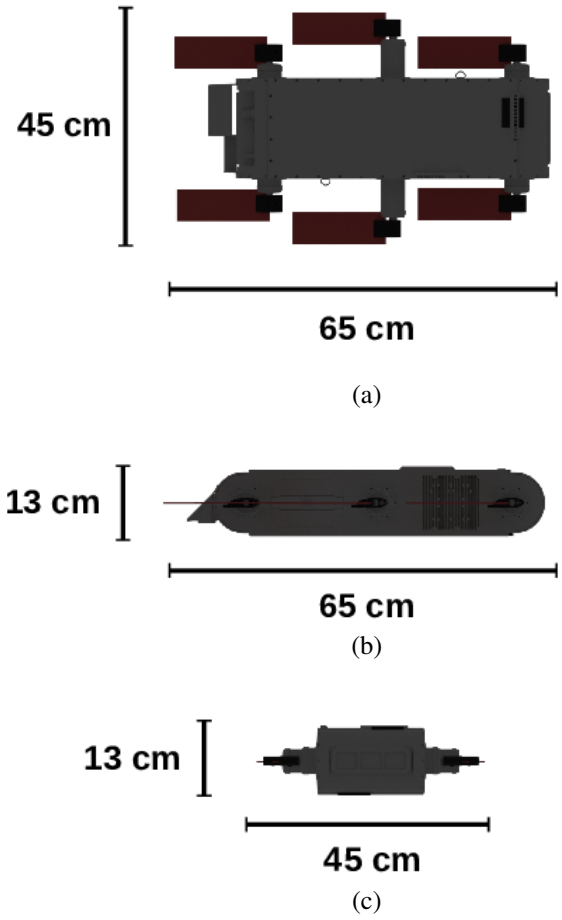
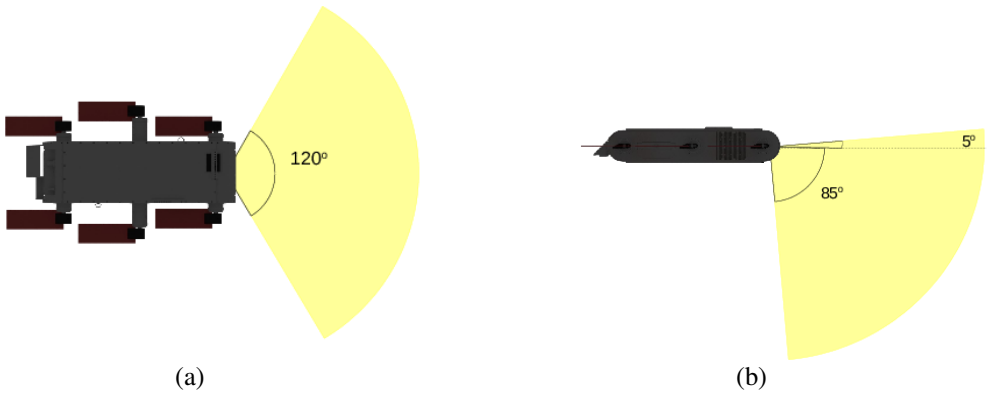


Figure 4.2 The dimensions of the Aqua2 from (a) top, (b) side , and (c) front perspectives.

4.2 Problem Statement

4.2.1 System Overview

The primary target system of AquaNav is the Aqua2 amphibious platform [107], pictured in Figure 4.1. Aqua2 is an amphibious hexapod robot, approximately 65cm $\times$ 45cm $\times$ 13cm in size and 10kg in weight, Figure 4.2. In its aquatic configuration, Aqua2 uses the motion from six flippers, each independently actuated by an electric motor, to swim. Aqua2 has 6 degrees of freedom, of which five are controllable: two directions of translation (forward/backward and upward/downward), along with roll, pitch and yaw.



(a) (b)

Figure 4.3 (a) The top and (b) the side view of the FOV of the Aqua2 highlighted in yellow.

The robot's computational system consists of two units, one responsible for vision and high-level planning and the other responsible for control related computations Each of these computers is based on dual-core Intel Core i3-6100U CPU @ 2.30GHz running Ubuntu 16.04 LTS.

The robot's primary sensing modality is vision [18]. It is equipped with three iDS USB 3.0 UEye cameras: two facing forward and one in the back. The front-facing cameras are used for navigation and data collection, whereas the back camera is used for communication with the operator using AR tags to send commands [108]. In addition to these cameras, Aqua2 also has an IMU and a pressure sensor which are used for controlling the motions and can be utilized for visual-inertial state estimation [8, 109–111]. The fields of view of the cameras are 120 degrees (horizontal) and 90 degrees (vertical) tilted downward by 40 degrees from the horizontal plane, as shown in Figure 4.3.

An important challenge of underwater robotic systems is navigating with currents and other fluid dynamics phenomena that can threaten the success of the mission. Regarding the Aqua2, which has a propulsion system of unique capabilities but also unique complexity, previous work [103–105] provided some analysis on the dynamics of the Aqua vehicle. However, the model is still inaccurate and, moreover, the on-board computing resources do not allow for utilizing a physics-based simulator online.

To move the robot, a PD controller proposed by Meger et al. [112] is utilized, which employs the IMU and the depth sensor data. This closed-loop controller accepts commands in the form:

$$com = \langle v, h, d, q \rangle \tag{4.1}$$

in which v is the desired forward linear velocity, h is the desired heave (that is, upward or downward linear velocity), d is the desired depth to reach and q the desired orientation for the robot to move in the world frame. The controller, while performing relatively stably and robustly, possesses oscillations that are present which should be mitigated during planning.

4.2.2 Problem Definition

Let the robot's pose be described using the vector

$$s = \begin{bmatrix} x & y & z & q_w & q_x & q_y & q_z \end{bmatrix}, \tag{4.2}$$

in which $p_s = [x, y, z]$ represents the position of the robot in the world frame and $q_s = [q_w, q_x, q_y, q_z]$ contains the coefficients of the unit quaternion specifying the robot's orientation. Also let s_{init} indicate the initial pose of the robot, s_{goal} the desired goal, O a collection of polytopes defined by a set of geometric primitives representing the observed or known 3D obstacles of the environment, PC_s for a state s the polytope defined by a set of geometric primitives g representing the volume the robot occupies in state s, and d_{safe} a desired clearance. The goal is to produce a trajectory $S = [s_{init}, s_1, s_2, \ldots, s_{n-1}, s_{goal}]$ so that:

$$\min_{g \in \bigcup_{s \in S} PC_s} \min_{o \in O} |g - o| \geq d_{\text{safe}} \tag{4.3}$$

The trajectory should guarantee also that the desired clearance is kept also during the transitions. Let the set of all transition states between two consecutive states s_{i-1} and s_i, produced by continuous linear interpolation denoted by $L_{s_{i-1}}^{s_i}$. The volume $L_{s_{i-1}}^{s_i}$ could be described as the convex hull of $PC_{s_{i-1}} \cup PC_{s_i}$. Then for every $s_i \in S - \{s_{init}\}$:

$$\min_{g \in L_{s_{i-1}}^{s_i}} \min_{o \in O} |g - o| \geq d_{\text{safe}} \tag{4.4}$$

In other words, a trajectory is needed to be produced from s_{init} to s_{goal} so that for every state on the trajectory, a clearance of at least d_{safe} is maintained. Then the motion commands generated to follow the above trajectory should not oscillate the robot more than d_{safe} from the original planned trajectory. Figure 4.4 shows an example of the motion planning problem with the basic elements introduced above.

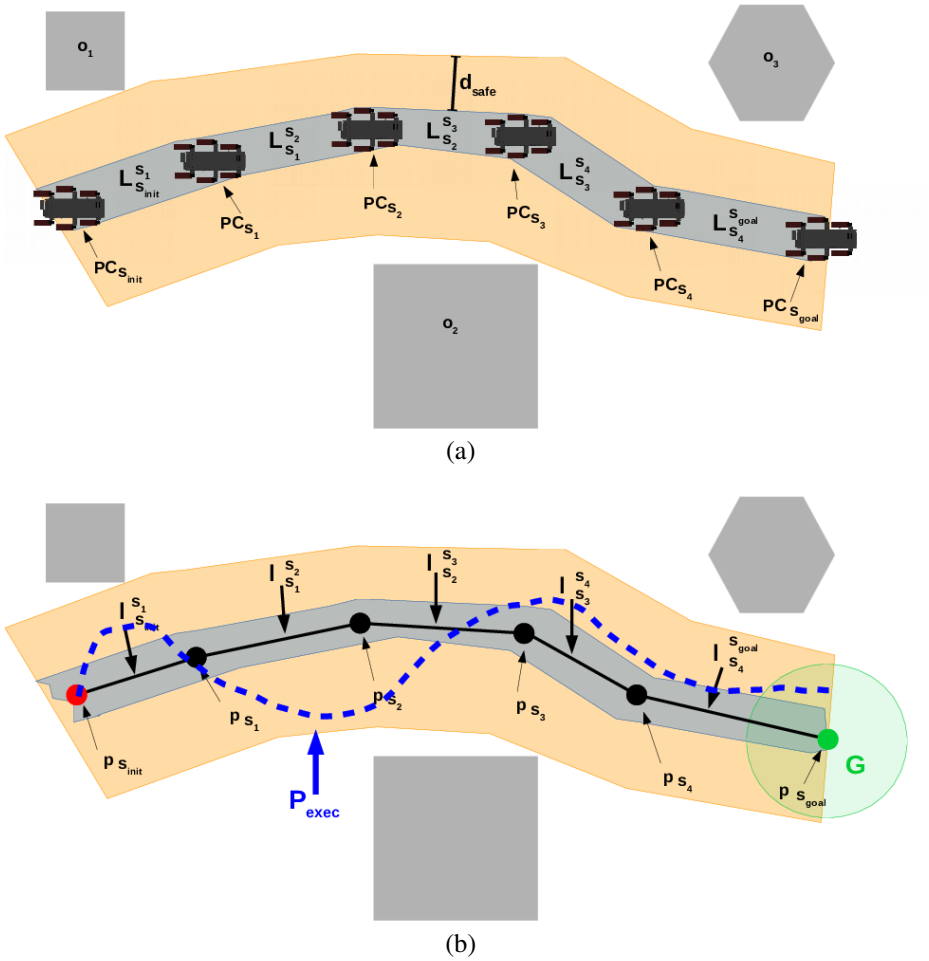


Figure 4.4 An instance of the navigation planning problem (a), and with an executed trajectory (b).

More formally, let P_{exec} denote the set of all states of the continuous path executed by the robot and R_{goal} a goal radius that defines a ball G with center $p_{s_{goal}}$. The goal is achieved and the path is considered potentially satisfiable if and only if:

$$\{s \in P_{exec} | p_s \in G\} \neq \emptyset \tag{4.5}$$

Let D_r indicate the distance from the assumed center of the robot C_r to its furthest point:

$$D_r = \min_{g \in PC_{s_0}} |g - C_r| \tag{4.6}$$

The executed path P_{exec} is considered safe if and only if $\forall s \in P_{exec}$ where the robot transitions from s_{i-1} to s_i with $s_i \in \{S - \{s_{init}\}\}$, we have:

$$\frac{|(p_s - p_{s_{i-1}}) \times (p_s - p_{s_i})|}{|p_{s_i} - p_{s_{i-1}}|} < d_{\text{safe}} - D_r \tag{4.7}$$

assuming the initial and goal configurations do not violate the desired clearance. The first part of the above equation calculates the euclidean distance of the point p_s from the line connecting two consecutive states $l^{s_i}_{s_{i-1}} = \{p_s | s \in L^{s_i}_{s_{i-1}}\}$; then the trajectory P_{exec} is considered to be valid and the robot reaches the goal successfully while avoiding the obstacles safely.

Finally, a planned path is considered valid if it's potentially satisfiable and safe, so if and only if Equation 4.3 holds and the oscillations from the original trajectory are bounded during execution for any P_{exec} as stated in Equations 4.5 and 4.7. For our navigation purposes, we are interested strictly in valid paths, since they guarantee both reaching the goal successfully and safely avoiding obstacles.

4.2.3 Discussion on the Assumptions

Less rigorously, by producing valid paths during motion planning, the solution guarantees that for every waypoint and transition, if followed precisely without any oscillation, the robot will have a distance of at least d_{safe}. Then, if the robot, during execution, remains no further than $d_{\text{safe}} - D_r$ from the planned path, the robot will reach the goal avoiding any collisions.

As stated earlier, an accurate dynamics model has yet to be introduced for the Aqua2 platform, so by utilizing simply a PD controller [112] based on periodic gaits the system

often becomes unstable and oscillates. A way to practically overcome this issue is to account for the maximum expected oscillation, given a certain speed, and achieve safety by guaranteeing a minimum clearance larger than the maximum expected oscillation during the planning process. A value for this maximum expected oscillation can be found experimentally during field trials and by using offline physics based simulators.

Focusing on the Aqua2 platform when operating in challenging environments, assuming a valid goal, robust state estimation, clear visibility of the obstacles, and a valid value of a maximum expected oscillation, the navigation problem could be reduced successfully to path planning with a desired clearance d_{safe}. This assumption, although arguably sounding reductive by dismissing the true complex dynamic model of the robot, led to superior navigation capabilities as tested in pool trials, and in challenging high-current conditions during open-water deployments.

In the case that at least one of the above assumptions does not hold, there is no guarantee that the robot will safely and successfully reach the goal. Especially regarding the importance of the desired clearance and how it affects planning due to a worst-case orientated policy that applies to the entire trajectory, it is true that solutions that the robot might be able to execute safely will be discarded, and this can lead to inability to find any valid solutions in very cluttered environments, though they might exist. However, this is a necessary trade-off between safety and completeness due to the absence of a sufficient dynamics and kinematics model for the robot.

Last but not least, notice that Equations 4.3 and 4.4 are describing a strict and very conservative approach. Paths that might partially violate the above conservative constraints might lead to superior performance and ways to rank and pick the best solution that violates the least these constraints could be considered. In the experiments the robot is shown to be able to perform robustly also in very cluttered environments with narrow passages, in which clearance is partially violated.

4.3 Related Work

In environments with complex dynamics, such as underwater, one of the main challenges is to generate safe paths. Real-time underwater navigation remains a fundamental, yet to be addressed sufficiently problem [113]. Several methods have been explored to correct the deviations caused by inertia and currents, including the FM* planning system [114]. Other methods rely on observations about the structure of the terrain [115] and satellite imagery [116] to estimate the effects of currents. Genetic algorithms [117] and mixed integer linear programming approaches [118] have also been used to support the computation of paths in dynamic underwater environments.

The work of Hernandez *et al.* [119] provided an online sampling based framework for an AUV in 3D, accounting also for the dynamics of the system. The proposed framework, although able to work in real-time, needed close to half a minute for producing solutions with clearance guarantees. Moreover, during the online experiments the replanning was close to 1Hz but the robot was constrained to a constant depth reducing the planning space to 2D. When accounting for currents, other studies [120, 121] utilize sampling-based techniques and a complete dynamic model of the AUV, but are only shown to work in uncluttered environments with few obstacles.

Another challenge in underwater path planning is to generate paths rapidly enough to be able to compute and execute them online. Green and Kelly demonstrated a branching-based method for quickly generating safe paths in a 2D environment [122] which has since been the basis of several optimizations [123–126]. Path planning has also been optimized by reducing many candidate paths to equivalence classes [127].

Though optimal sampling-based techniques [3, 42–44] provide near-optimal solutions and have improved over time, they, in general, require more computational resources, more time, and often an exhaustive search of the configuration space. Some studies on online underwater navigation with sampling-based techniques quickly generate safe paths, how-

ever, they are limited to 2D motions [128] or require additional assumptions regarding the vertical relief [129] without exploiting the full potential of available 3D motions.

In some applications, it is necessary for AUVs to navigate in an environment without global knowledge of the environment. In such cases, obstacles are observed, often by stereo vision as has been done on aerial vehicles [130]. Exploration of an unknown environment by aerial vehicles has been performed using a 3D occupancy grid using probabilistic roadmaps (PRM) and the D* Lite algorithm for planning [131]. Although underwater and aerial domains provide different challenges, both require path planning in 3D. For an AUV such as Aqua2, whose movements do not correlate exactly with control inputs, planning becomes even more difficult. Other AUVs have also been used for path planning, such as RAIS [132] and DeepC [133]. Another AUV, REMUS [134], used obstacle avoidance specifically for exploration of shallow waters.

The Aqua2 AUVs have a variety of swimming gaits in order to enable tasks such as swimming in a straight line, on the side, in a corkscrew motion, or performing a barrel roll [135]. Visual tags were used to enable the AUV to navigate over structures [112]. Furthermore, the robot learned a reactive controller for navigating over the coral reef while maintaining a safe distance [136, 137].

4.4 Method Description

The objective of this work is to develop algorithms that enable the Aqua2 robot to navigate reliably and safely through a dense field of obstacles, given start and goal poses s_{init} and s_{goal}. Figure 4.5 presents an overview of the proposed system.

4.4.1 Trajectory Planning

Motion planning in this context must balance several competing constraints, including the need for efficiency, the possible need to replan, and the limited computational power available on the robot (particularly when one considers the other essential tasks of perception,

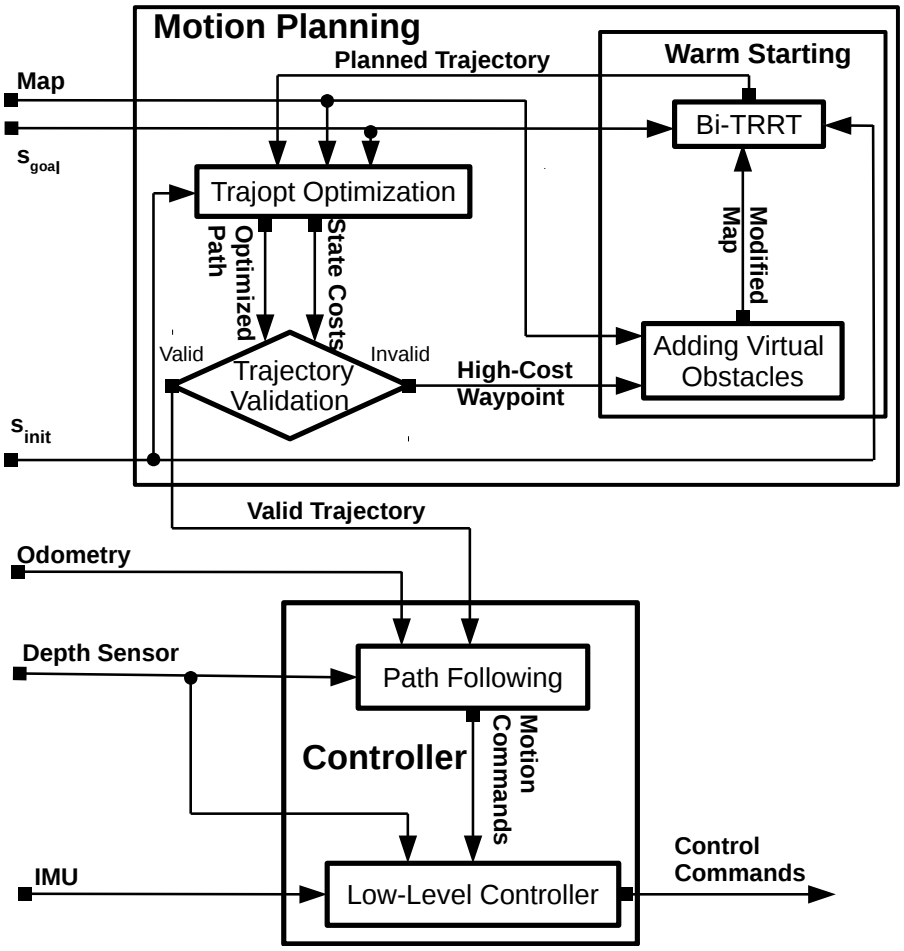


Figure 4.5 System architecture.

mapping, etc.). In addition, because of its complex —yet not readily modeled— dynamics and kinematics combined with the unpredictable nature of maritime currents, the system is quite susceptible to disturbances. Thus, the planner should provide solutions that satisfy some minimum clearance to avoid collisions.

To satisfy this challenging trade-off, the proposed system utilizes an optimization-based planning approach. Specifically, the implementation uses Trajopt [7], which has been proven as a very robust method for manipulators and mobile manipulators in 2D. It is computationally efficient and takes into account not only the states but the complete path between them using the transition between states, contrary to alternatives such as

CHOMP [4] and STOMP [54]. To the best of authors' knowledge, Trajopt has not yet been used for 3D motion planning for mobile robots, nor in the underwater domain, in the past.

The main idea behind Trajopt is to represent the path S from s_{init} to s_{goal} as an ordered list of waypoints, each of which is a pose for the robot. Starting from an initial set of waypoints —generally based on the linear interpolation between s_{init} and s_{goal}— Trajopt forms a convex optimization program, in which each degree of freedom of each waypoint is a variable, the obstacles are encoded as constraints, and the objective is to minimize a weighted form of the path length. An important advantage is that, because of this general form, one can insert additional content-specific constraints, such as a maximum depth.

As mentioned in Chapter 2, Trajopt is a very robust and efficient path optimization package, capable of solving fast challenging motion planning problems. In a nutshell, using the notation introduced before, Trajopt attempts to minimize the following function:

$$f(S) = \min_{S} \sum_{i=1}^{n-1} ||s_{i+1} - s_i|| \tag{4.8}$$

where $S = \{s_0, s_1, \ldots, s_n\}$ the set of n states considered during optimization. The above equation is the sum of square displacements that minimizes the path length.

Then collision constraints are enforced for every state $s \in S - \{s_0, s_n\}$:

$$h(s) = \sum_{o \in O} |d_{\text{safe}} - sd(PC_s, o)| \tag{4.9}$$

where O is the set of obstacles, PC_s the 3D volume of the robot in state s, and sd a singed distance metric between 3D objects. More details for the sd function can be found in the original Trajopt paper [5].

The above constraint, given successful convergence, guarantees that each state will keep at least d_{safe} from the closest obstacle, but has no guarantees on the in-between transitions. In order to enforce continuous-time safety, instead of applying Equation 4.9, the following function is applied for each $s_{i-1}, s_i \in S$ consecutive states:

$$H(s_i, s_{i+1}) = \sum_{o \in O} |d_{\text{safe}} - sd(L^{s_i}_{s_{i-1}}, o)| \tag{4.10}$$

where as defined before, $L^{s_i}_{s_{i-1}}$ is the convex hull of $PC_{s_{i-1}} \cup PC_{s_i}$. For the purpose of AquaNav, an additional constraint was added to avoid surfacing shown at Equation 4.11.

4.4.2 Input Methods

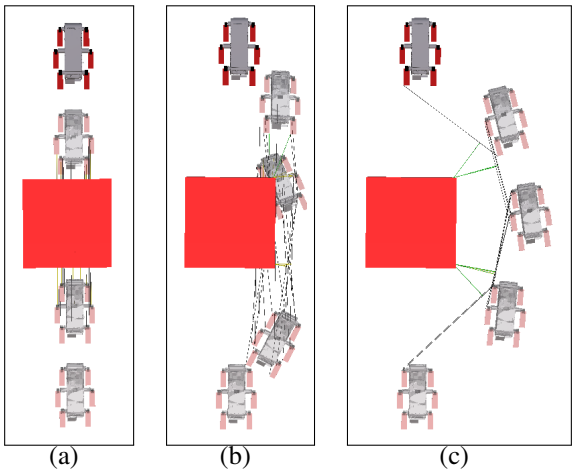
Trajopt optimizes the distance between the swept-out volumes of the robot's trajectory and the obstacles; as shown in Figure 4.6. For efficiency, Trajopt expects the map to be stored in a form where the normal vectors between the robot's body and the obstacles can be extracted rapidly. One geometric method for presenting the obstacles to Trajopt, suitable in cases where the environment is well-known, consists of specifying the obstacle shapes and locations as instances of a set of built-in primitives, typically as rectangular boxes.

We also implemented a version that utilizes a point cloud input representation, which can be produced directly from raw sensor data. First, the point cloud is provided as input to the fast surface reconstruction algorithm of Zoltan et al. [138]. Then, the algorithm approximates the produced mesh via a collection of convex polytopes which can be directly processed by the Trajopt optimizer.

4.4.3 Constraints

The objective function is parameterized by coefficients for the path length and the obstacle avoidance and by a distance parameter d_{safe}, measuring the maximum distance from the obstacles where the cost will be applied. These parameters required tuning for the underwater environment and the AUV used; Section 4.5 describes the specific values utilized in our experiments.

Our system employs an additional term in the cost to ensure that the robot will not reach the surface, and will remain entirely underwater. The cost function c_z was applied on the z



(a) (b) (c)

Figure 4.6 Path optimization with Trajopt [7] in three different stages: (a) The initial path is generated by simple interpolation from s_{init} to s_{goal} with possibly some states in collision. (b) An intermediate stage during optimization. (c) The final trajectory, shown with the distances to the swept-out volumes.

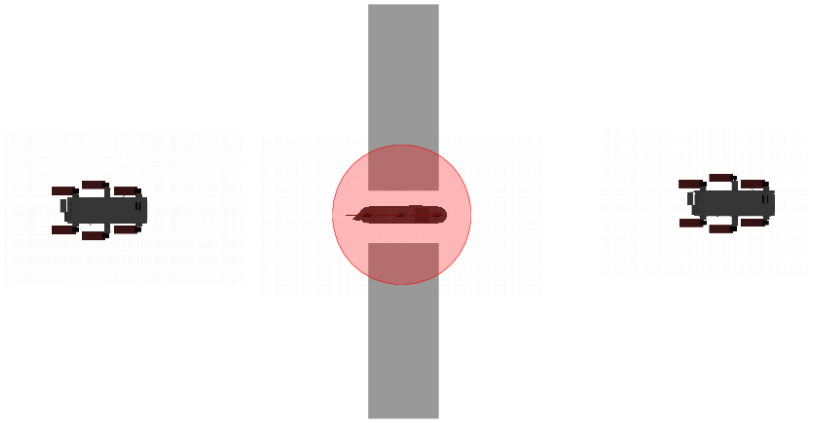
coordinate of all the states $s_i \in S$, and defined as follows:

$$c_z(z_i) = \begin{cases} z_i + \varepsilon & \text{if } z_i > -d_{\text{safe}} \\ 0 & \text{otherwise} \end{cases} \tag{4.11}$$

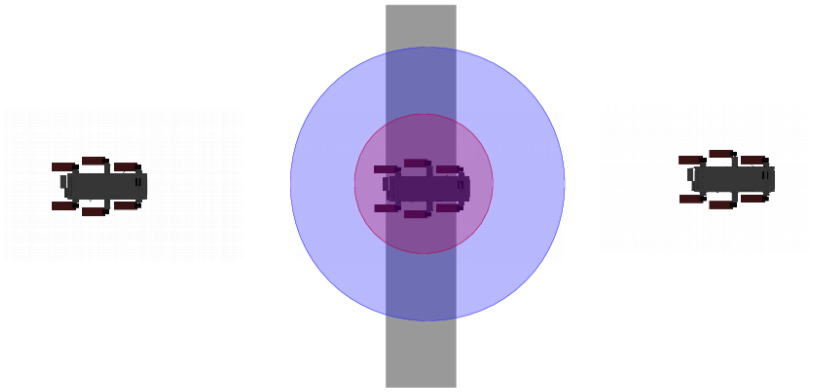
where $\varepsilon > 0$. Assuming that on the surface $z = 0$, the condition in Equation 4.11 penalizes every state above $-d_{\text{safe}}$ ensuring that the robot will remain underwater, accounting also for potential inaccuracies in control.

4.4.4 Overcoming Local Minima

A problem that many optimization-based motion planners, including Trajopt, face is the possibility that the optimization may converge to a local minimum. Though generally rare in the implemented system, this situation can present a safety hazard for the robot, because the completed path may not necessarily maintain safe distance from the obstacles. Local minima can be present, for example, either (a) when the path is passing through an obstacle and there is no free space in a d_{check} radius that Trajopt is using to check for obstacles, or (b) when the path is passing through a narrow passage and the desired clearance d_{safe} from the



(a)



(b)

Figure 4.7 Failure cases of the Trajopt optimization. In (a) the optimization is stuck and the robot passes through a narrow passage violating the clearance d_{safe}(light red). In (b) the robot passes through an obstacle that is larger than the d_{check} (light blue).

obstacles cannot be maintained. As shown in Figure 4.7, there are two different situations in which optimization fails due to local minima issues:

1. The path passes through a big obstacle where no free space exists in a radius d_{check}, or

2. The path passes through a narrow passage that violates the desired clearance.

Fortunately, it is straightforward to verify whether or not the output of Trajopt does indeed avoid the obstacles correctly, by checking for collisions or large cost values. Typi-

cally, such issues are resolved with warm-starting of the optimization process: a fast motion planner is used to generate beforehand a set of valid paths and these paths are used consecutively to initialize the optimization until a valid solution is found. An example of such work [139] for Trajopt uses the BiT-RRT [140] planner.

We propose the following novel *iterative* warm-starting approach, using BiT-RRT [140], by inserting virtual obstacles into the planning process. The optimization step works with the motion planner in the following manner:

1. The optimization result is checked and in case of failure, the waypoint with the highest cost is selected.

2. The map used by the motion planner is altered by adding a virtual obstacle of size d_{safe} in the position of that waypoint.

3. The planner plans a path avoiding the new high cost area of local minimum with a smaller than d_{safe} distance from obstacles during collision checks.

4. The new path is used to initialize the path optimization process and the procedure continues if needed.

This process forces the planner to identify an alternative path that avoids the most problematic portion, in terms of the cost function, of the previous path. An example execution of this procedure is shown in Figure 4.8.

4.4.5 Trajectory Following

Localization

The problem of underwater localization has proven to be extremely challenging [20,22] due to the lighting variations, hazing, and color loss [141]. We tested two solution strategies for this problem.

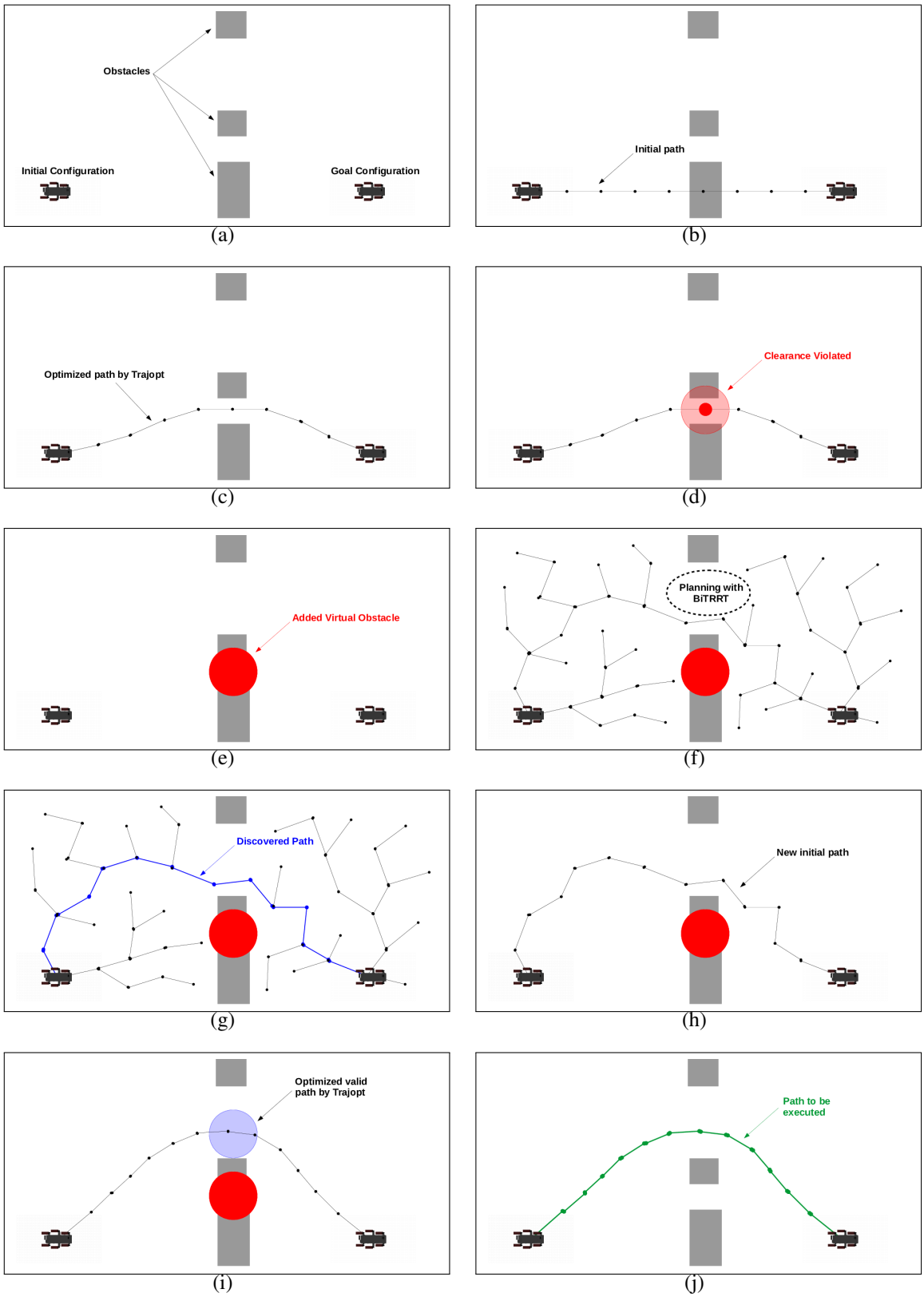


Figure 4.8 An example of the warm-restarting procedure.

Primitive Estimator A primitive estimator has been employed using the depth sensor, the attitude estimation from the IMU and the expected forward speed of the AUV (based on the swimming pattern utilized). Though subject to drift over long distances, this estimator has guided the Aqua2 vehicles for a variety of basic maneuvers and swimming patterns. During operations with a known map, this primitive estimator is utilized to track the planned trajectory.

SVIn During operations in an unknown environment, more accurate localization may be useful, in addition to the required ability to detect nearby obstacles. Visual Inertial state estimation, introduced by Rahman *et al.* [8] and later extended to utilize depth measurements and loop-closures [142], is used for simultaneously localizing and mapping nearby obstacles. The resulting point-cloud produced by SVIn enables the AUV to navigate safely around obstacles.

WAYPOINT SEEKING

The optimization stage of the proposed pipeline produces in a timely manner a path p, as an ordered set of consecutive goal states that should be sequentially achieved by the robot. Then, the PD controller, with control commands as shown in Equation 4.1, is utilized to follow the path. The current framework, for simplicity considers only purely forward motion setting $h = 0$.

The PD controller controls the desired depth by linear interpolation. Assuming that the current state of the robot is s_c and the current i goal position is p_i in the world frame, to guarantee smooth transitions that are bounded inside the calculated safe swept-out volumes of the optimization stage, the desired depth d is calculated as

$$d = z_{p_{i-1}} + \left(1 - \frac{z_{s_c}}{z_{p_{i-1}}}\right)\left(z_{p_i} - z_{p_{i-1}}\right), \tag{4.12}$$

where $z_{p_{i-1}}$ is the depth of the previous achieved goal (base case $z_{p_{i-1}} = s_{\text{init}}$), z_{p_i} the depth of the current goal, and z_{s_c} is the current measured depth. The idea is to ensure the linear

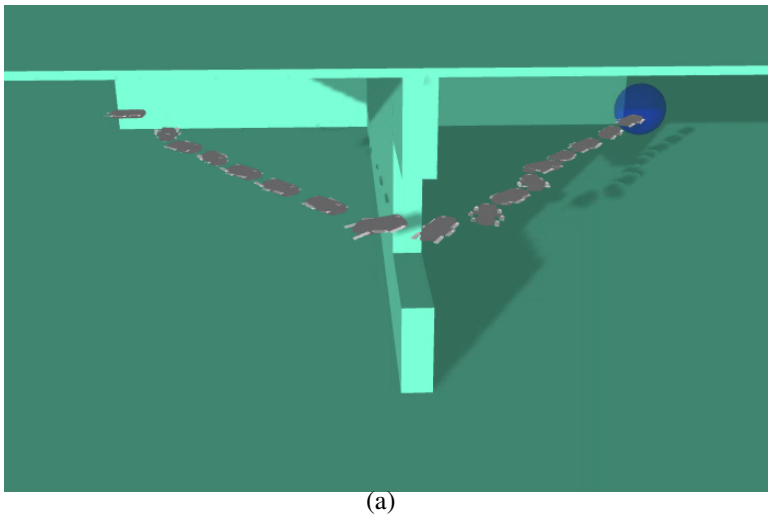
change of depth from one position to the other and ensure linear transitions similar to the ones assumed by Trajopt.

Regarding the desired orientation, the pitch is adjusted automatically from the desired depth and the roll does not affect the direction of the motion, thus only the computation of the desired yaw o_{yaw} is needed, with respect to the translation error $e_t = p_i - s_c$. Given the position of the robot, the yaw changes in such a way that the AUV will always move towards facing the goal.

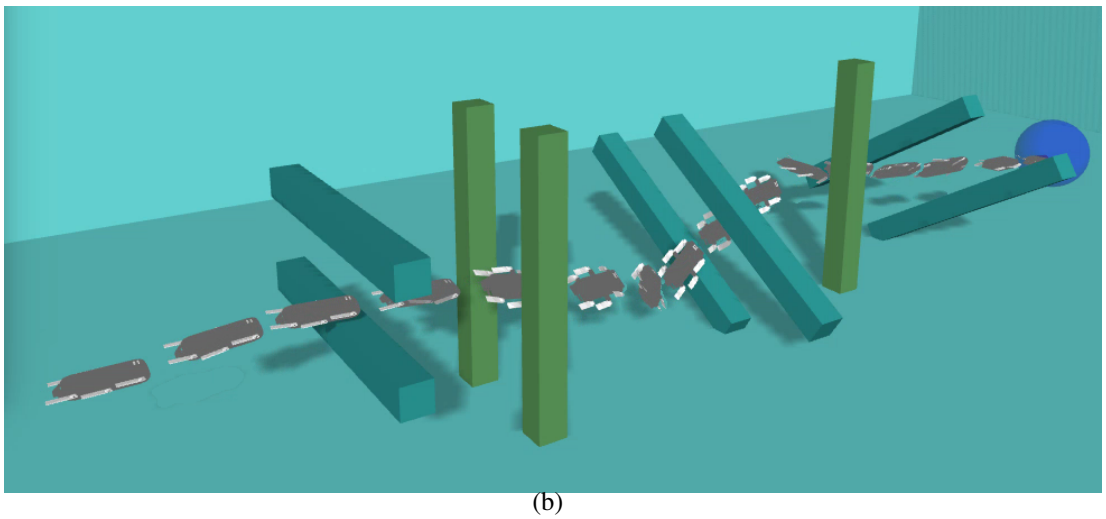
Lastly, assuming the possible deviation is bounded by d_{safe} for a given speed and that the optimization was successful, the AUV should safely navigate from one goal to the next one. As a result of the above, given a threshold of d_{reached}, the goal is declared reached, if the error e_t is less than d_{reached} and a local minimum is detected, since otherwise the robot will be sliding away due to disturbances.

4.5 Evaluation

Extensive experiments were performed using the Gazebo simulator, and in numerous deployments of the Aqua2 AUV at two pools in our university, a shallow swimming pool of dimensions $50m \times 25m \times 2m$ and a deep diving pool of dimensions $25m \times 15m \times 4m$. The main objective of the various experiments was to demonstrate the reliable navigation functionality of Aqua2 using the proposed framework. In all of the experiments the AUV had a constant speed of $0.4m/s$ — the expected operational speed — and a bounded motion with a minimum obstacle avoidance distance of $d_{\text{safe}} = 0.6$m. Obstacle avoidance and path length coefficients were adjusted to relatively high values 200 and 100, respectively, favoring safety over path length optimality, while the number of waypoints was determined either by the distance from the current position to the goal — placing a state every $1.5m$ — or by the number of states provided by the Trajopt planner. The experiments tested planning on a known map, with a focus on efficient trajectories, and online, using the camera for obstacle avoidance and frequent replanning.



(a)



(b)

Figure 4.9 Simulated trajectories executed by the robot in Gazebo. (a) An environment with a narrow opening (the ceiling is not shown); (b) A cluttered environment with multiple pipes.

4.5.1 Known Map — Offline Planning

During planning with a known map, the input map was a set of geometric primitives (planes, boxes, cylinders, etc.) and the iterative warm-starting process was used. All plans were produced in less than half a second.

SIMULATED ENVIRONMENT

Two different environments are presented here highlighting different challenges for Trajopt. In the first environment, called the Window environment, two rooms, separated by a wall with an opening (window) between them and open to one side demonstrates path optimality; see Figure 4.9(a). The original path results in a local minimum. After the iterative warm-starting method is used the solution through the window is found. Furthermore, for this experiment the AUV has to keep a horizontal pose during motions that causes larger drifting from inertia. For this purpose, in this single case we increased d_{safe} to 1m.

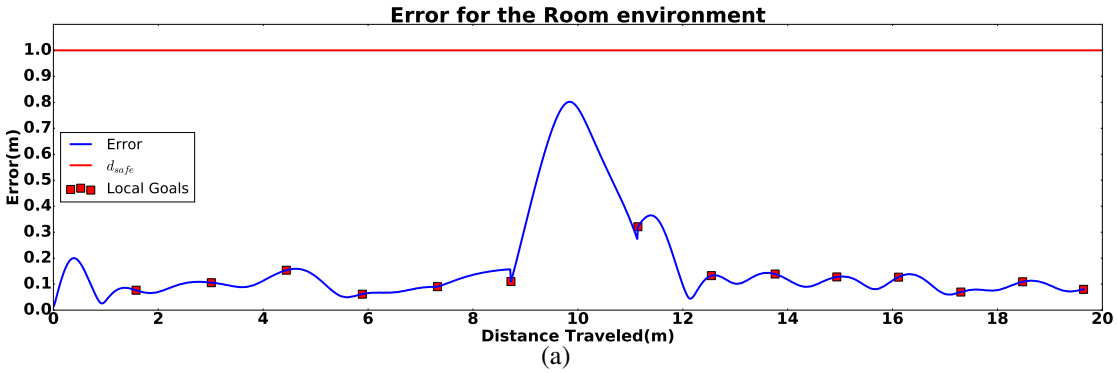
The second environment, called the Cluttered environment, focuses more on the capability of our method, inherited from Trajopt, to minimize oscillation while passing through a sequence of obstacles that need accurate motions with fast orientation changes; see Figure 4.9(b). In this environment not only was the roll adjusted during optimization adapting to the motion, but also the Aqua2 is guided to pass in a parallel motion between each pair of pillars, thus maximizing the distance from both of them, for increased safety.

Safe navigation for the Aqua2 is achieved only if during the motion the trajectory following method does not violate the assumed clearance d_{safe} by the planning process. For these challenging scenarios, where the robot changes orientations fast, the oscillations are shown in Figure 4.10. The error at time t, e_t, is calculated as the Euclidean distance of the measured simulation odometry s_c, to the line formed by the previous p_{i-1} and the current local goal p_i:

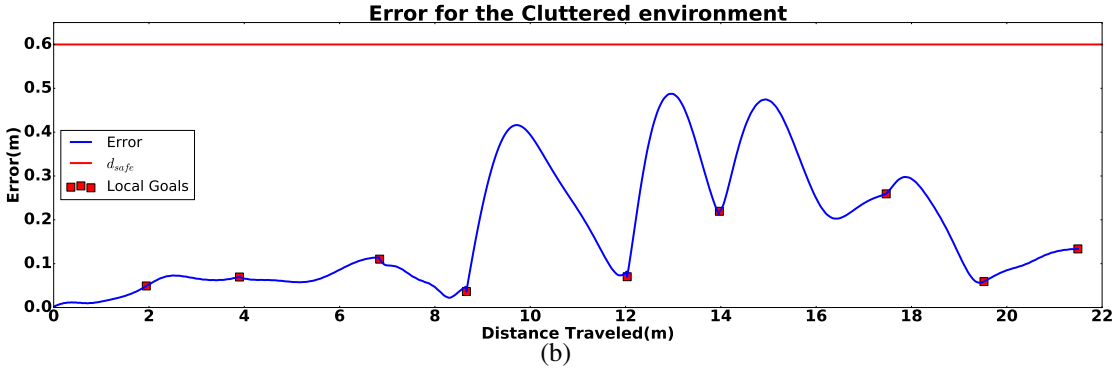
$$e_t = \frac{|(s_c - p_{i-1}) \times (s_c - p_i)|}{|p_i - p_{i-1}|} \tag{4.13}$$

POOL TRIALS

Four different environments were used at the two pools in our university, demonstrating operations with the Aqua2 AUV; see Figures 4.13, 4.14, 4.15, and 4.16. The placement of the obstacles followed the created map used as input to the augmented Trajopt planner.



(a)



(b)

Figure 4.10 Error diagrams for the Room (a) and the Cluttered (b) environment, as measured from the simulation. The red squares indicate the local goals achieved, and the red line marks d_{safe}, where errors larger than that should be considered unsafe.

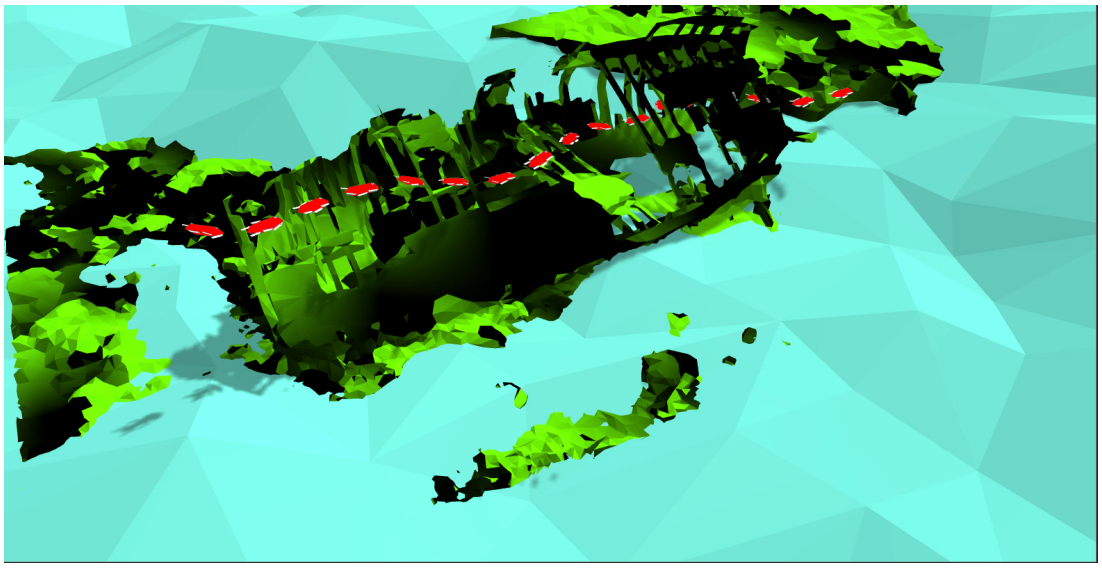
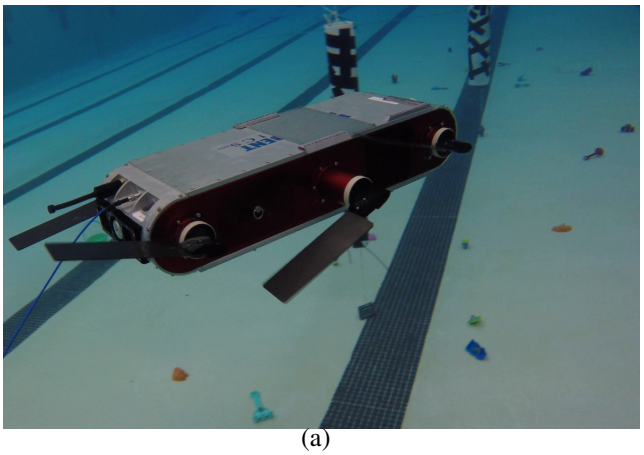
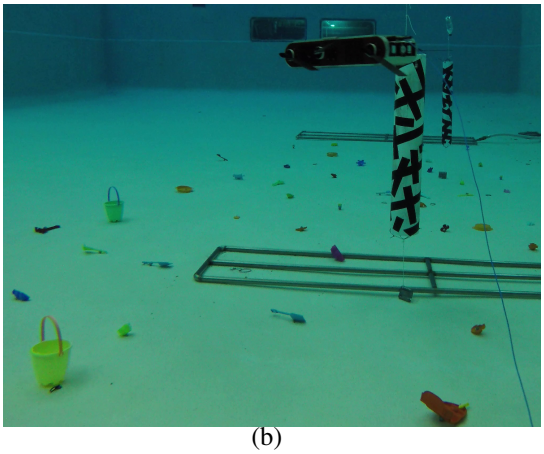


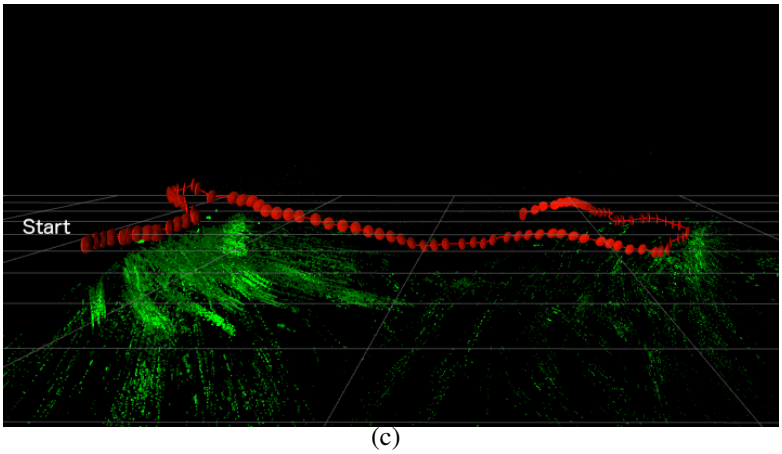
Figure 4.11 Navigating inside a shipwreck model in simulation. Model of "Shipwreck, Hooe Lake, Plymouth" from Sketchfab.



(a)

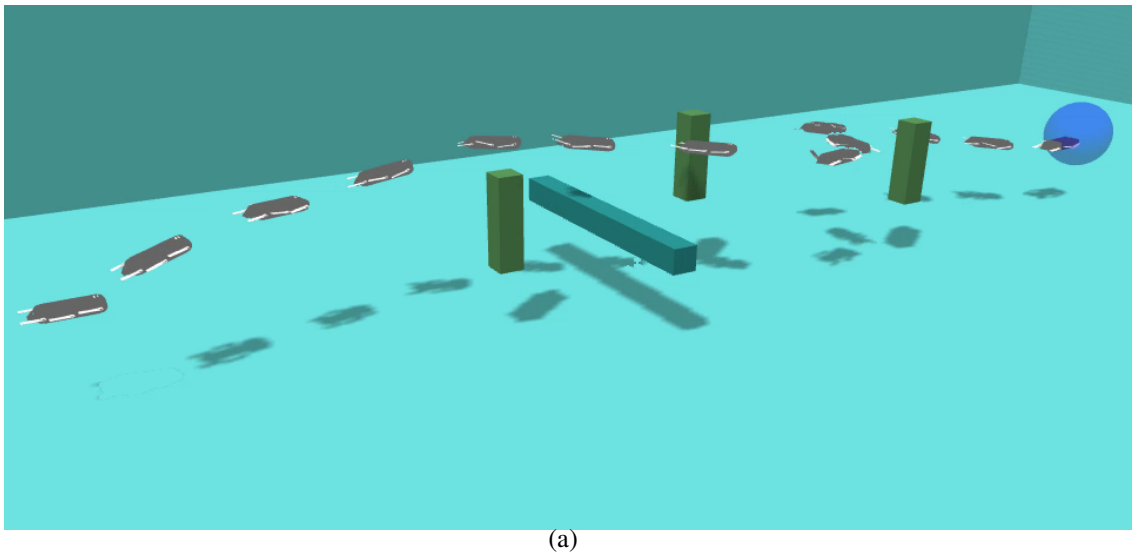


(b)

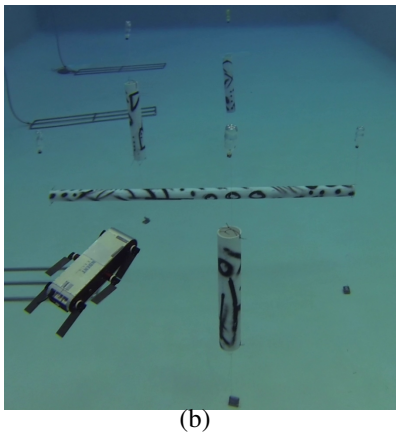


(c)

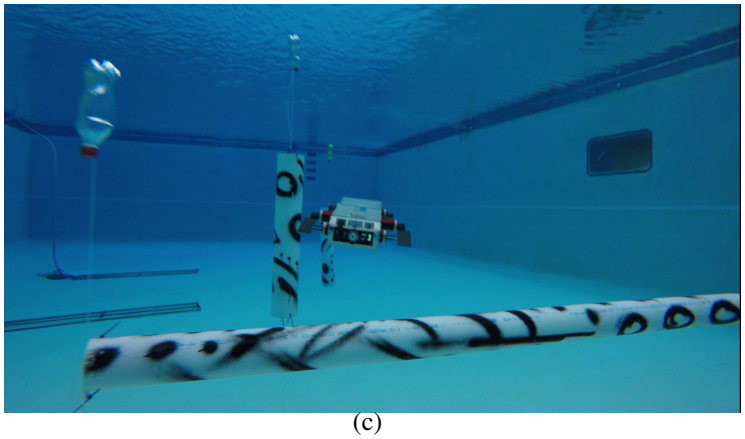
Figure 4.12 (a),(b) show representative photos from the deployments in the pool in an unknown environment. Please note the plastic toys spread at the bottom of the pool to produce detectable features in a featureless pool. (a) Avoiding two obstacles in the shallow swimming pool; (b) Avoiding two obstacles in the deep diving pool. (c) presents the online map produced by SVIn [8] as a screenshot of RViz for the environments of (b), the robot avoids the first cylinder, moves forward and then avoids the second, while using the features from the bottom of the pool to localize.



(a)



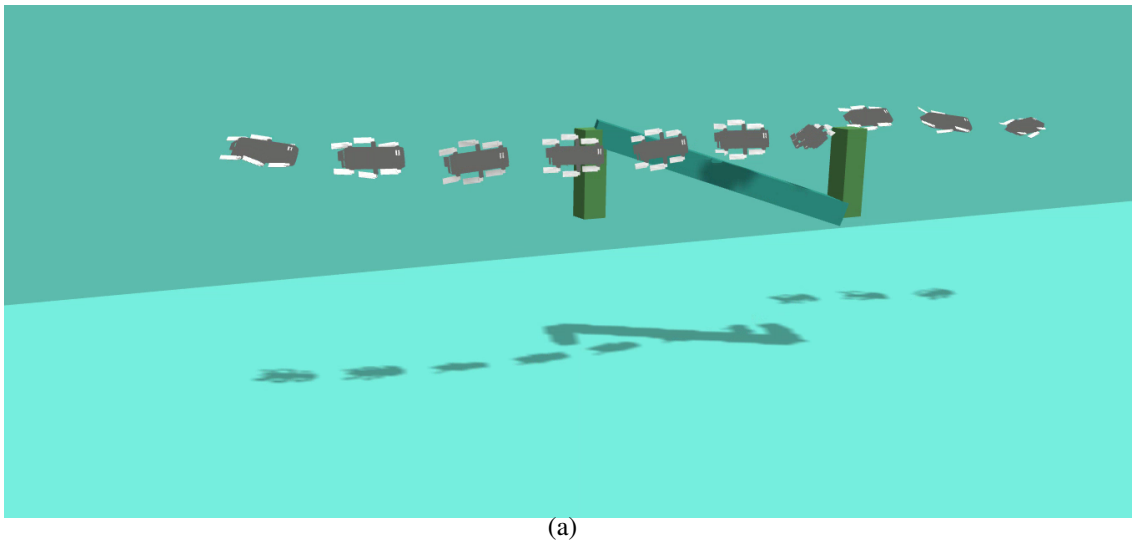
(b)



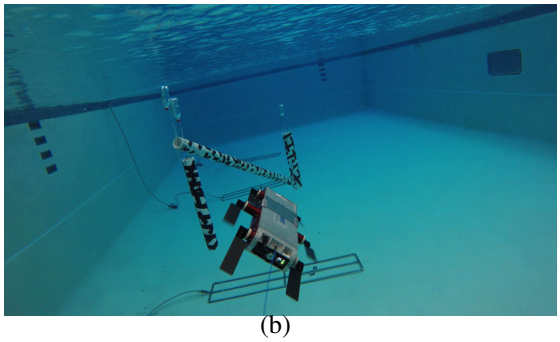
(c)

Figure 4.13 A trajectory of the simulated environment (a), and 2 snapshots of the in-pool executed trajectory at (b) and (c). The robot was forced to keep constant roll orientation to test a bottom monitoring behavior.

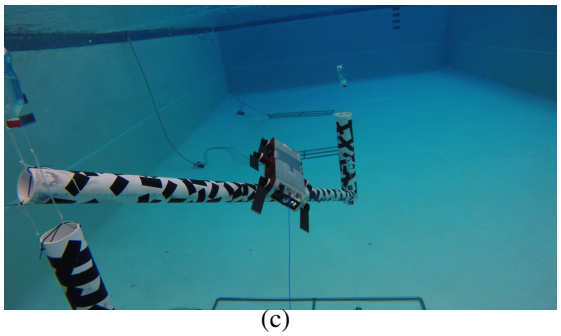
In the first environment, Figure 4.13, the Aqua2 was forced to maintain an attitude facing the floor, a situation that maximizes drift during yaw changes. The robot was able to successfully avoid all the obstacles and quickly self-correct its orientation. For the other three environments, the attitude of the AUV was optimized by the planner. In the second environment, Figure 4.14, the robot avoided the two vertical pipes and passed over the diagonal one. Three horizontal pipes were used to force the AUV in continuous depth changes; see Figure 4.15. Finally, three vertical pipes resulted again in obstacle avoidance with a roll at 45°; see Figure 4.16.



(a)



(b)

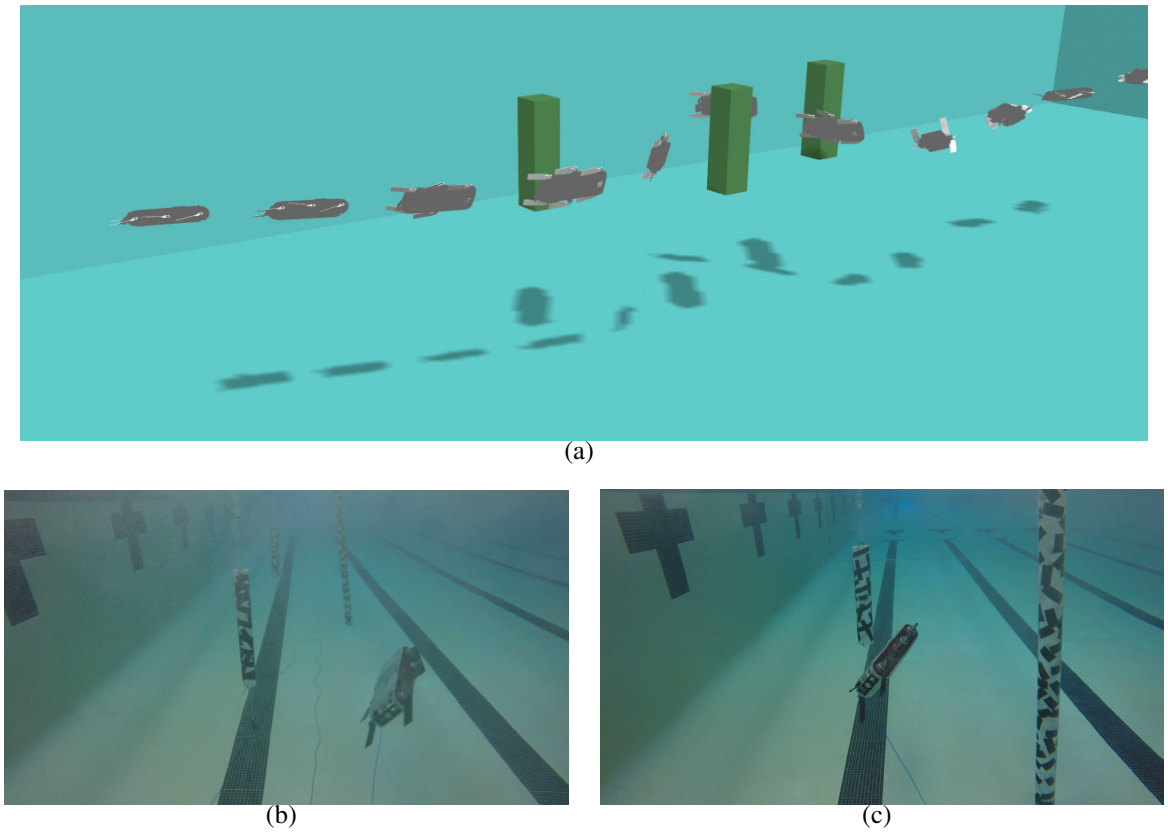


(c)

Figure 4.14 A trajectory of the simulated environment (a), and 2 snapshots of the in-pool executed trajectory at (b) and (c).

4.5.2 Sensor based Planning — Online

The deployment of the proposed framework online highlighted some computational challenges. First, the state estimation consumes a large fraction of the computing resources. Second, the most computationally expensive component of the motion planning pipeline is the convex decomposition step, using approximately $70-80\%$ of the total planning time. For efficiency, the convex decomposition parameters were adjusted to produce many small convex polyhedra, instead of a few large ones. A history of the observations was kept to account for the limited field of view of the AUV. In all cases, the replanning frequency was on average at 1Hz.



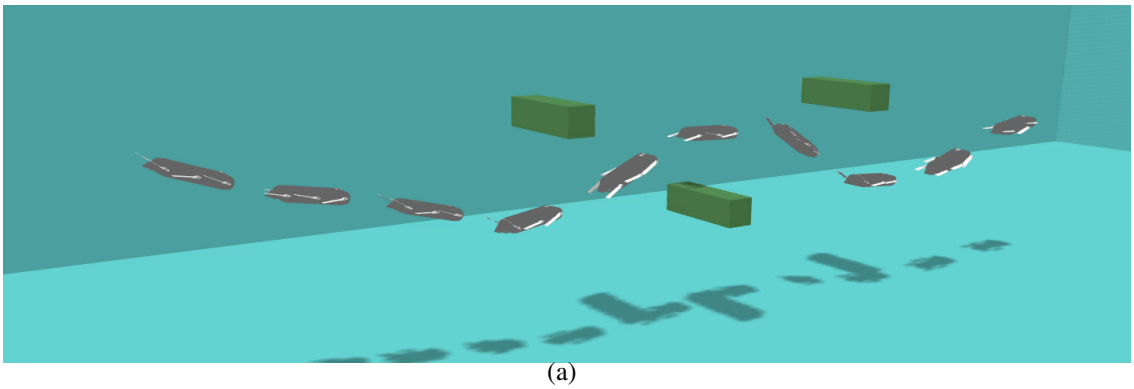
(a) (b) (c)

Figure 4.15 A trajectory of the simulated environment (a), and 2 snapshots of the in-pool executed trajectory at (b) and (c).

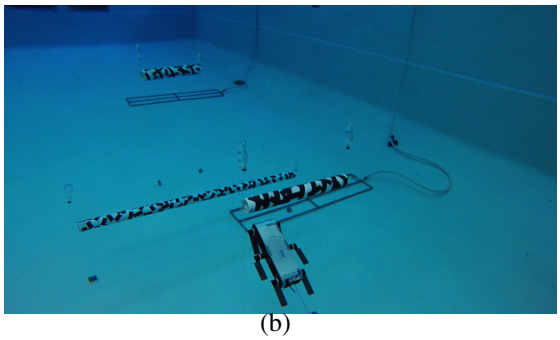
SIMULATED ENVIRONMENT

The model "Shipwreck, Hooe Lake, Plymouth" from Sketchfab[1] was simplified and used in the Gazebo simulator. For computational efficiency, no texture was used in the simulator and the basic point-cloud was acquired from the model using a resolution of 100×75 points. The AUV was guided through the inside of the shipwreck, while avoiding the observed obstacles; see Figure 4.11. The complex environment presented in Figure 4.9(b) was used for online navigation resulting in similar results.

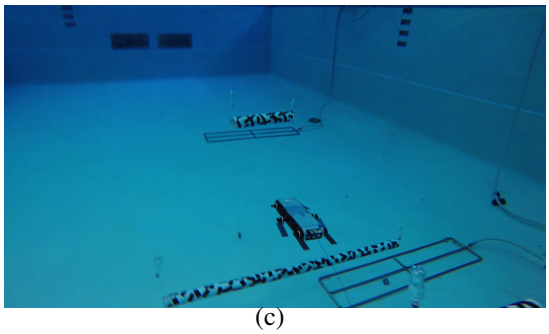
[1] `http://sketchfab.com/`



(a)



(b)



(c)

Figure 4.16 A trajectory of the simulated environment (a), and 2 snapshots of the in-pool executed trajectory at (b) and (c).

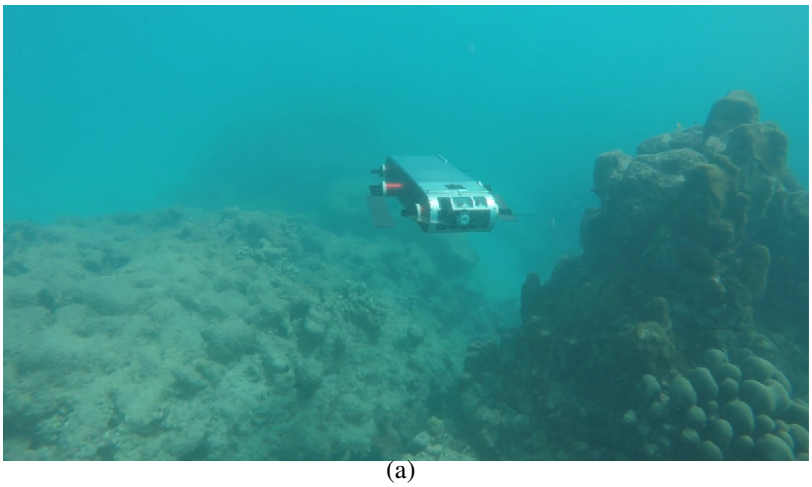
POOL TRIALS

Additional real pool experiments were conducted using state estimation from the robust underwater SLAM package SVIN [142] with additional sand-toys weighted and placed on the floor to improve the odometry estimation together with obstacles to test obstacle avoidance; see Figure 4.12. The same configuration was used with the online simulation framework with the difference that the AUV was constrained at constant depth to maintain tracking using the features on the floor. The two environments, at the shallow and deep pool, used two vertical obstacles resulting in similar obstacle avoidance trajectories. Figure 4.12(c) presents the overall recorded trajectory and features detected from SVIn [142]. During one of the experiments the AUV's motion brought it towards a diver recording the experiment, who, to the diver's relief, was treated as another obstacle and was avoided.

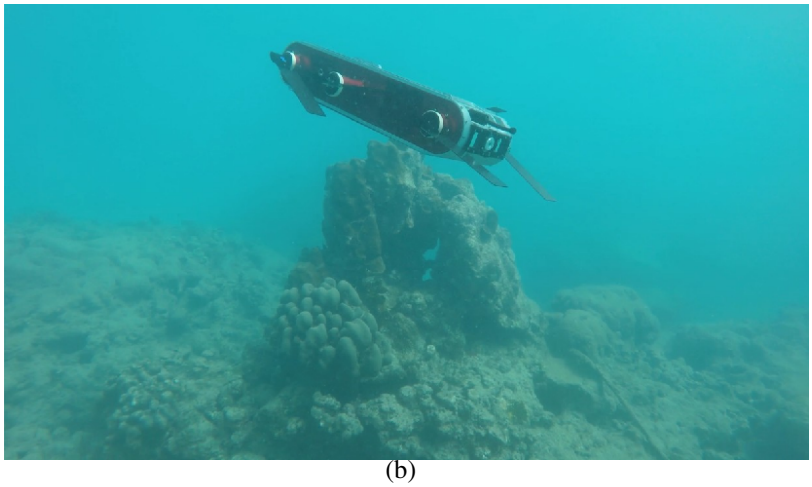
OPEN WATER DEPLOYMENTS

Finally, open water deployments were conducted last January, at the Bellair's Research Institute of McGill University, in Holetown, Barbados. The experiments aimed to test the online capabilities of AquaNav in open-water challenging conditions, on the left for constant depth and roll and on the right for moving unconstrained in 3D.

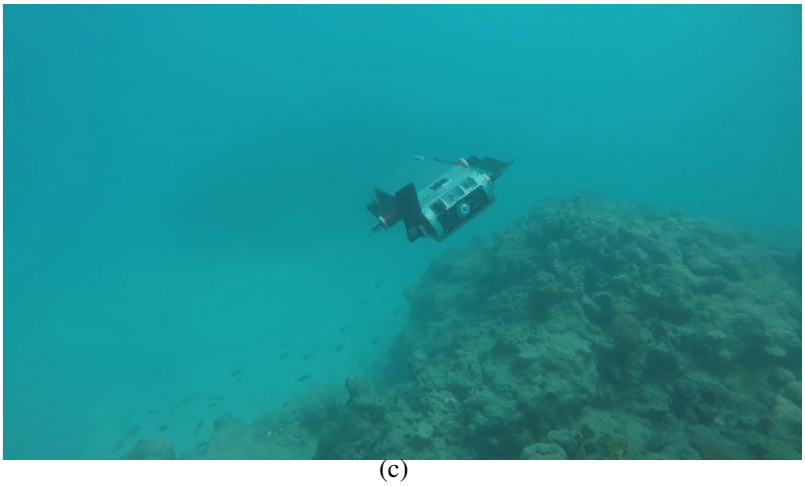
Given the very challenging conditions due to turbidity and high-currents, SVIn was assisted by dead-reckoning. In all cases AquaNav was able to reach the goals and successfully avoid static or dynamic obstacles such as divers or schools of fish. As shown in Figure 4.17 three different behaviors were tested in order to emphasize the modularity and the adaptive nature of the framework. For the first experiment, the robot was forced to remain at constant depth and no roll was allowed. For the second the robot was able to change depth and no roll was allowed. Then, the robot was fully utilizing its 3D kinematic capabilities. In all cases, despite the high currents the robot kept safe distance from all obstacles and safely navigating to the predefined goal.



(a)



(b)



(c)

Figure 4.17 Aqua2 using AquaNav navigating over a coral reed in open-water trials, for (a) constant depth and no roll, (b) variable depth and no roll, (c) fully 3D locomotion.

CHAPTER 5

AQUAVIS: PERCEPTION-AWARE AUTONOMOUS NAVIGATION FOR UNDERWATER VEHICLE

5.1 INTRODUCTION

Underwater operations using Autonomous Underwater Vehicles (AUVs) is a research topic that attracts attention of an increasing number of researchers and organizations in both academic and industrial settings. Pushed by recent advancements in hardware, real-time state estimation, and motion planning techniques, this strong current towards realizing autonomy in the underwater domain is supported by many potential applications, such as marine archaeology, underwater infrastructure inspection and maintenance, energy and resource utilization, public security, and environmental monitoring. Moreover, due to climate change threatening maritime infrastructure and requiring constant monitoring in isolated and hard to reach marine environments, underwater autonomy is becoming more essential than ever.

Currently, most —if not all— essential underwater tasks are performed by human operators directly who risk their health or even lives, or remotely by controlling ROVs which require significant human resources and logistics. In the first case, even excluding the risks, operations are constrained by water conditions, and especially the hard limitations on maximum depth and operation duration set by human biology. In the second case, even excluding the deployment costs, tethered operations are limited to unconfined spaces and generally uncluttered environments. Also, the human operators are controlling the ROV based on limited information from its sensors, which potentially leads in delays and

underutilization of the platform's capabilities due to very conservative and overcautious operation.

Autonomous Underwater Vehicles (AUV) could perform tasks underwater without additional motion, depth, and duration limitations and especially without any risk to human lives. But AUVs deal with other important issues that are raised in the underwater environment, both with hardware and software. Especially, autonomous underwater 3D navigation for agile robots using vision-based state estimation, and motion uncertainty, is very challenging for various reasons. A major bottleneck of underwater autonomy is robust vision-based SLAM [20, 22], leading many state-of-the-art platforms to rely completely on dead-reckoning in order to avoid the visibility challenges of the underwater domain such as color attenuation, turbidity, and lack of color saturation, illumination, and feature-rich areas. A second important bottleneck is real-time motion planning, which should deal with motion uncertainty and safe operations in proximity to underwater structures and cluttered environments for monitoring, mapping, and exploration purposes.

Previous work has addressed these issues providing robust solutions for visual-inertial-based underwater state-estimation, by introducing SVIn [8] and SVIn2 [142], and a complete robust underwater navigation framework, called AquaNav [17], which was presented in Chapter 4. Though AquaNav provided safe and efficient paths that avoided obstacles in real-time, it had no consideration on the future visibility of the few feature-rich areas of the underwater domain, which SVIn2 and many other vision-based SLAM techniques rely upon. Additionally, for the same reasons, AquaNav lacks the awareness needed for inspection, mapping, and monitoring purposes, since often the way for the robot to avoid detected obstacles, involves motions that steer the robot away from the feature rich obstacles. For example, as shown in Figure 5.1 the robot might turn right in order to avoid the obstacles perceived on the bottom left corner, but such motion might drive the robot away from high-textured areas forcing the vision-based SLAM method to lose track. Thus, introducing a new methodology for solving the important problem of active perception [143] extends

autonomy by driving the robot towards feature-rich areas, for minimizing state-estimation uncertainty or for observing areas of interest.

In order to achieve such desired behavior, it is highly important to combine perception and motion planning, in order to avoid the obstacles, but also keep feature-rich objects in the cameras' field of view. Bringing perception and motion planning closer not only assists state-estimation, but also produces trajectories that track and monitor points of interest, such as fish, corals, and structures. Such behavior is preferred for exploration and monitoring strategies that should collect diverse and meaningful-to-humans information [144, 145].

Figure 5.1 An example robot view of the Aqua2 while operating underwater. The red frame highlights feature-rich areas, while yellow areas with few poor-quality features. AquaNav might drive away from the red area Aqua2 to avoid collisions, but such maneuver might result to loss of view of these essential to state-estimation features.

This chapter proposes a novel formulation for active perception and a novel framework for a real-time perception-aware underwater navigation framework, called AquaVis [19]. The proposed pipeline builds on the existing AquaNav pipeline and enables an underwater robot with an arbitrary multi-camera configuration to perceive multiple visual objectives, extracted automatically, along the path for mapping, monitoring, or localization purposes, by introducing two novel cost-functions in the optimization process. The objective of AquaVis is to extend the capabilities of AquaNav [17] so that motions will be gener-

ated enabling the robot not only to move efficiently and avoid obstacles safely, but also to observe objects of interest while performing these motions, Figure 5.2. These visual objectives can be perceived online and from a desired distance, while at the same time the robot safely reaches the desired goal. Though the primary focus lies on multi-camera configurations, as shown in the experiments, other sensors such as sonars and LIDARs, or combinations of sensors of different capabilities and attributes, could be used within the proposed approach to offer more robust performance.

Observing objects of interest rather than being our primary focus — which is the view of the typical coverage problem [146–149] — it is considered a secondary objective with the primary being the efficient and safe navigation to the specified goal position. In the ideal case a solution would be a sequence of motions where if followed by the robot, the goal is reached, any collisions are being avoided, the path length is kept minimal and during any transition at least one point of interest is being observed during at each state.

The contributions of AquaVis, upon successful completion, will be:

1. A system able to safely navigate through cluttered challenging environments; an inherited property from AquaNav [17].

2. A perception-aware method for a forward moving robot that is capable of tracking multiple visual objectives along the path.

3. A navigation scheme able to track objects from a desired proximity mitigating turbidity.

4. A high-level local planning technique, that allows modifications, adjustments, and exhibits sophisticated performance.

5. Experiments validating the performance of the method in different scenarios and for different multi-sensor configurations.

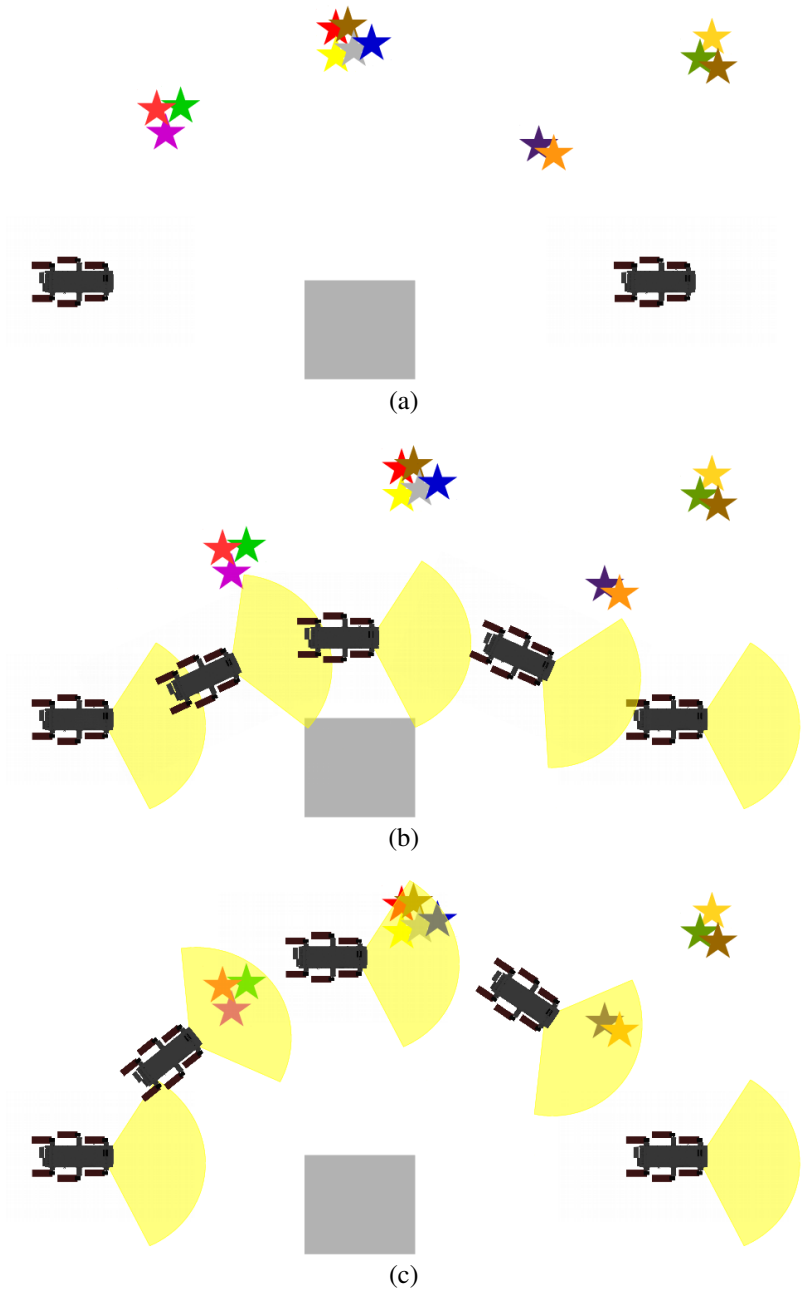


Figure 5.2 An instance navigation around obstacles (grey) and feature-rich visual objectives indicated with stars (a). AquaNav, might emphasize on avoiding obstacles and minimize the path length, ultimately potentially losing track (b), while AquaVis attempts to safely navigate by avoiding obstacles and at the same time let the robot observe some nearby visual objectives.

5.2 Problem Statement

Building upon the formulation of AquaNav described in section 4.2.2, let the set of v objects to be potentially observed denoted by V, and F_s denoting the visibility manifold at

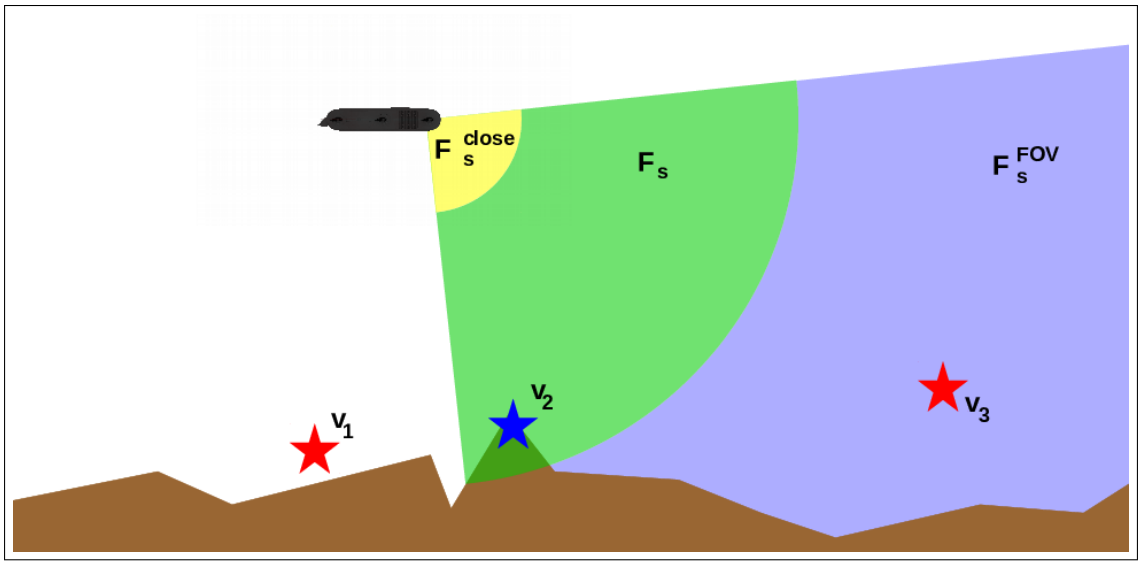


Figure 5.3 The visibility formulation of Equation 5.1. F_s^{close} is shown with light yellow, F_s^{FOV} with light blue and the visibility manifold F_s with light green for the state s of the robot. The visual objectives v_1, v_2, and v_3 are indicated with stars. Only v_2 is visible because it is inside F_s, while v_1 and v_3 are not observed.

state s for a robot with a single camera, within the robot observes objects. Let also F_s^{close} be the sphere centered at the camera's focal point with radius r_{close}, before when nothing that close is considered visible, and similarly F_s^{far} the sphere with radius $r_{far} > r_{close}$ behind which nothing that far is considered visible at state s. If F_s^{FOV} defines the spherical sector of the field of view of the camera, then:

$$F_s = (F_s^{FOV} \cap F_s^{far}) - F_s^{close} \tag{5.1}$$

For a given state s of the robot and a point of interest $v \in V$ —also called visual objective— we say that the robot observes the point of interest if $v \in F_s$, Figure 5.3.

The same formulation could be generalized in robots carrying arbitrary multi-sensor configurations. If C is a set of sensors, and F_s^c represents the visibility manifold for sensor c with $c \in C$, computed by Equation 5.1, then the visibility manifold for a robot at state s could be provided by:

$$F_s = \bigcup_{c \in C} F_s^c \tag{5.2}$$

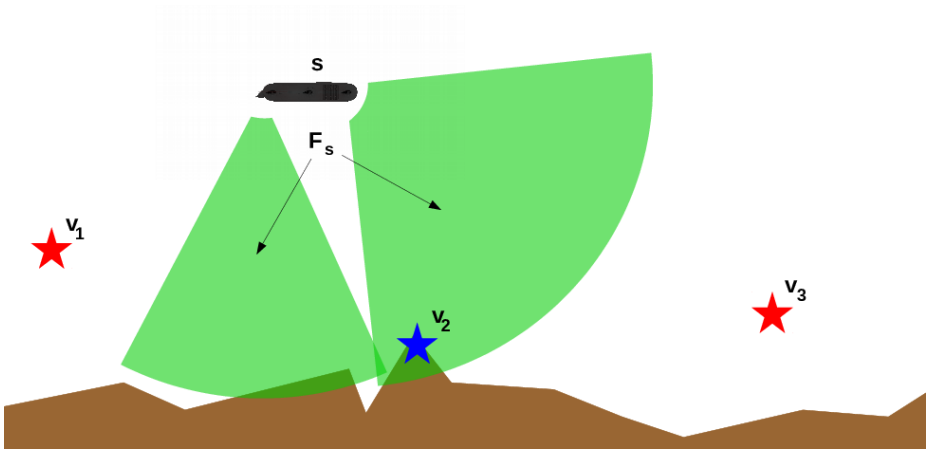


Figure 5.4 The visibility manifold F_s for 2 cameras mounted on the robot is shown in light green. Visual objectives v_1, v_2, and v_3 are indicated with stars. Only v_2 is visible because it is inside F_s, while v_1 and v_3 are not observable from the robot's current state s.

An example of the visibility manifold for Aqua2 utilizing both the forward looking stereo camera and a bottom-looking back camera is shown in Figure 5.4. It is worth noting that such formulation do not introduce any constraints on the type of the sensor or robot. Although not studied in the current dissertation, the same formulation is applicable to robots of arbitrary type, such as manipulators, mobile manipulators, humanoids, etc.

Finally, the goal is to maximize the number of states of the executed trajectory where the robot observes at least one visual objective. More formally, if P_{exec} is the executed continuous path of the robot approximated by $P^{\sim}_{exec} = [s_1, s_2, \ldots, s_n]$, where n approximates P_{exec} with arbitrary precision, then the path's success in maintaining visibility of the visual objects is quantified via the following function:

$$M(P^{\sim}_{exec}) = \frac{|\{s_i \in P^{\sim}_{exec} \mid F_{s_i} \cap V \neq \emptyset\}|}{n} \tag{5.3}$$

Equation 5.3 provides the percentage of the trajectory where at least one point of interest is visible. When $M(P^{\sim}_{exec}) = 1$ during the entire path the robot was observing at least one point of interest, while on the other hand when $M(P^{\sim}_{exec}) = 0$ the robot is unable to observe any points of interest.

Thus, the goal of AquaVis is to produce a path that minimizes the path length (Equation 4.8), under the continuous clearance constraints (Equation 4.10) similar to AquaNav, but additionally maximizing visibility of certain targets (Equation 5.3).

5.3 Related Work

The problem of active perception was first introduced by Aloimonos *et al.* [150] and Bajcsy [151], and then by Feder *et al.* [152] in the late 1990s, in the context of exploration. They introduced a framework that made local decisions to improve pose estimates during mapping based on uniquely identified landmarks. Stachniss and Burgard [153] provided a method that improved localization using SLAM, by attempting loop-closing.

Makarenko *et al.* [154], used a laser range scanner, extracted landmarks that were used with an Extended Kalman Filter, and they proposed a method that could be parameterized to trade-off exploring new areas with uncertainty. Martinez *et al.* [155] achieved reducing pose and map estimates with Gaussian Processes. The work of Rekleitis introduced a exploration vs exploitation based framework to reduce uncertainty for a single robot by visiting previously mapped areas [156], with extensions to using sonar sensors [157], and also multi-robot systems [158]. Zhang *et al.* [156, 159, 160] employed hybrid metric-topological maps, to reduce localization and mapping uncertainty. All these early works presented to this point, were considering only the 2D case.

More recent studies have expanded the problem from 2D to 3D, with the main platform considered being quadrotors, although few studies utilizing manipulators exist [161]. These studies are more related to our proposed objective, but in general they are not providing visibility distance constraints and unlike Aqua2 latteral motions and on the spot yaw rotations where considered. Other major differences will be highlighted.

The work of Forster *et al.* [162] provided a method to minimize uncertainty of a dense 3D reconstruction, but it was based on a direct method that have weak performance underwater due to turbidity, and mostly fly-over motions were performed without a robust

obstacle avoidance method. Penin [163] introduced a framework for producing trajectories taking into account the field of view limitations of the camera, but it was restricted to tracking only 4 points in close proximity to each other, no obstacles were considered, and computational expenses do not allow for real-time behavior.

The work of Spica *et al.* [164] combined visual servoing with Structure from Motion. They moved the camera with the objective of increasing the quality of the reconstruction. Discarding the difference regarding our goal of improving localization, instead of mapping, their method did not considered obstacles and operation in cluttered environments. Constante *et al.* [165] proposed to use a photometric method to drive the robot close to regions with rich texture and features, but similarly to [162] direct methods are not expected to perform well underwater and also the motions were constrained to fly-overs and near-hovering.

Sheckells *et al.* [166] provided an optimal technique for visual servoing, but no obstacle avoidance methods were introduced for cluttered environments and only one visual objective, consisting of features, was considered for the duration of the trajectory. But for a forward moving robot with a forward looking camera set, such as Aqua2, without the capability of lateral motions, in most cases more than one visual objective needs to be considered to keep track. Additionally, the works of Nageli *et al.* [167, 168] focused on visual-objectives tracking, rather than achieving a goal with robust localization, and the potential field method applied for obstacle avoidance could result in a local minimum in cluttered environments.

Other studies that considered only one visual objective are the following [169–173], and studies that did not consider obstacle avoidance are [165, 169, 174, 175]. It is worth noting that [175] indeed was able to track a set of landmarks, but with the constrained that they should be tracked always, while for our objective the robot needs to choose which objective should be tracked from the given position. [176] utilized a mounted camera on a gimbal and since it was based on a teach and repeat method, it is not applicable for

underwater unexplored environments. Given the camera configuration and the kinematics of the Aqua2, the study above, neither the method of Zhou *et al.* [177], nor Murali *et al.* [174] that allowed lateral motions and free on the spot yaw rotations could be applied for our purpose. Some techniques [170, 173] consider only one target, but also they resulted in low-level controllers, where our goal is to keep the planning component high level enough in order to encourage extensibility and complex and sophisticated performance. A very recent work by Zhang and Scaramuzza [178] proposed a new topological model for map representation that could be used for guaranteeing uncertainty reduction in the entire map, but a computationally expensive offline computation on a known map is needed before planning limiting the scope of online applications.

In the underwater domain Frolov*et al.* [179] a motion planning framework was proposed for reducing map uncertainty in the context of coverage for AUVs. Finally, there is a recent work by Manderson *et al.* [180], very closely related to the proposed objective, that applied a navigation framework on the Aqua2 platform. This technique was based on deep-learning fitting on data collected by a human operator controlling the robot. The robot was taught to stay close to corals, and avoid collisions with corals and rocks. Although that technique seems robust at first sight, it is unable to fully exploit the kinematic abilities of the robotic platform the way AquaNav does without considering roll motions, it is constrained to navigate only to similar environments (coral reefs), and the motion commands follow a very reactive behavior and a short decision window that was compensated in this work by following predefined local goals.

On the other hand, AquaVis produces locally near-optimal motions for avoiding the obstacles, with no reliance on a potentially error-prone human training process. It also produces efficient trajectories for safe navigation in cluttered environments, similar to AquaNav. More importantly, since it operates on point-clouds, localization could be maintained with any kind of structures with rich texture, without the limitations dictated by a training dataset. Moreover, it is able to incorporate third-party object recognition modules for

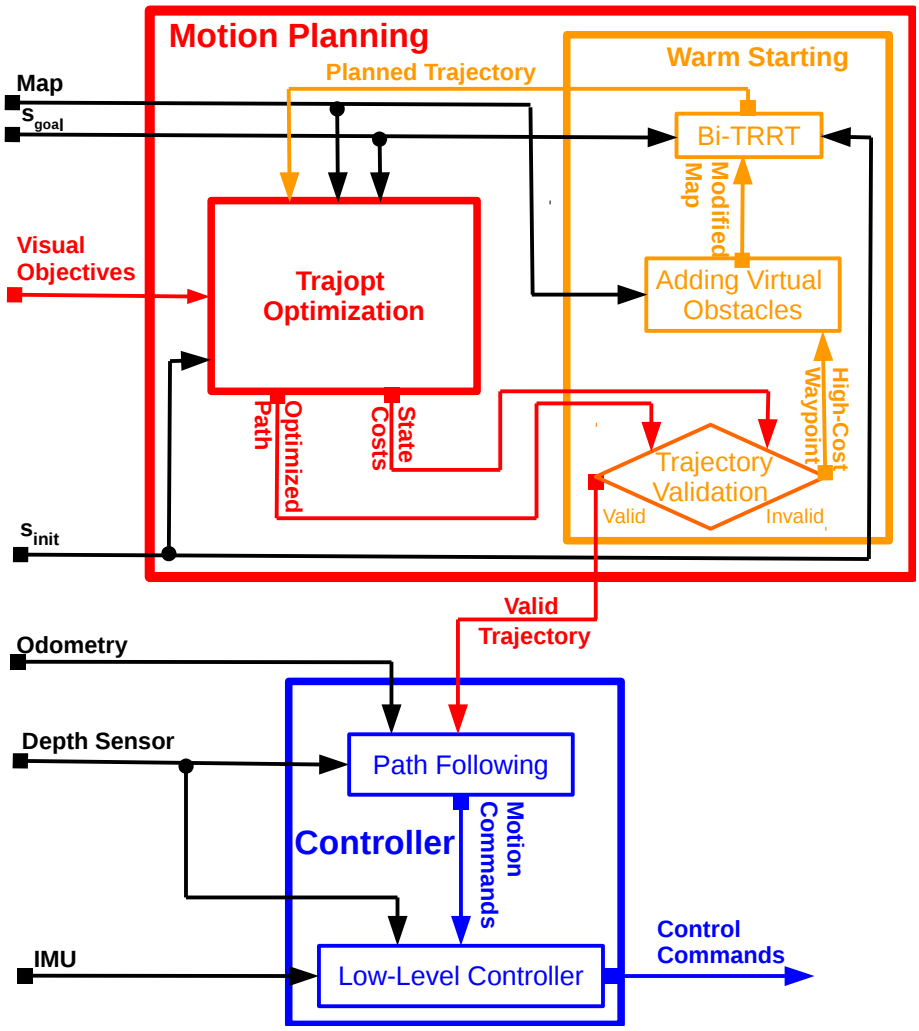


Figure 5.5 System architecture of AquaVis, which is based on AquaNav. AquaVis alters the core planning component by incorporating visual objectives, shown with red, while modules for warm-starting, shown with orange, and path following, shown with blue, are kept the same.

monitoring objects of interest, without the need of the time and resource intensive training on the motion planning module. Finally, it is applicable to robots of arbitrary multi-sensor configurations and enables them to navigate and observe multiple visual objectives along the path from a desired proximity.

5.4 METHOD DESCRIPTION

AquaVis, as an extension of AquaNav, inherits the same principles and guarantees. As described in Chapter 4, AquaNav has both on-line and off-line capabilities. It is able to navigate the robot safely while performing efficient and minimal paths in challenging scenarios. AquaVis, as a robust perception-aware navigation framework should also share all these attributes, along with producing paths that satisfy additional visibility objectives.

The proposed enhancements appends the AquaNav pipeline, on the point-cloud processing step, and the motion planning module, maintaining the state-estimation and path following steps. Such modularity further highlights the extendability of the original framework. A diagram of the proposed pipeline in Figure 5.5, highlights that the new modifications.

With respect to the point-cloud processing step, new functionality is implemented to detect objects with dense features and automatically extract visual objectives. Regarding the motion planning component, new cost-functions are developed, respecting the kinematics of the robot, and the capabilities of the path follower, in order to steer the paths towards the specified visual objectives.

The next paragraphs provide further discussion on these enhancements.

5.4.1 EXTRACTING VISUAL OBJECTIVES

AquaVis has both offline and online capabilities. For the offline planning, a known map is given, typically consisted of geometric primitives, similarly to AquaNav. In such case, AquaVis additionally requires the visual objectives to be observed as 3D points.

Fore online navigation, AquaNav employs the robot to navigate an unknown map, using a capable state estimation package such as SVin [181], that outputs both the odometry and a representation of the sensed environment as a 3D point cloud. During the online version of AquaVis, both the obstacles and the visual objectives are detected in the 3D space in real-time. Prior work of the AFRL lab has provided techniques for detecting corals [182–184],

or other Aqua2 robots [9] using CNNs. In this chapter, the primary focus is improving odometry estimates, instead of tracking of known objects. Thus, the raw point-cloud is processed to extract visual objectives with high density of features, and then these visual objectives could be used to assist the odometry as landmarks if the camera is able to track them.

A way to extract such objects is to treat the problem as a density-based clustering problem. Efficient density-based approaches, such as DBSCAN [185], could be utilized on the point cloud to omit clusters with high density and then the centroids of these clusters is chosen as the visual objectives.

DBSCAN [185] is highly parameter free, since only a minimum neighboring distance and a minimum number of samples per cluster is needed to be tuned. Regarding the first parameter the minimum dimension of the robot is used (30cm). For any distance of two points smaller or equal to 30cm, the points are treated as points representing the same obstacle, given that the robot will not be able to fit in-between. The second parameter is adjusted to a low value, given our observations and experience [17] that highlight the scarcity of good features underwater in comparison to other environments. A value from 5 to 10 seemed to produce acceptable results in datasets collected from areas the robot is expected to operate.

DBSCAN [185] is a very powerful clustering technique and it is capable of detecting highly non-convex clusters. This is a very desired attribute for most clustering problems, but in the current application it contradicts with the requirement of extracting only convex clusters, so the centroid will lie within the high feature density regions. Although specific counterexamples could be constructed as failure cases for the proposed approach, to our experience, such cases are extremely uncommon underwater, since rocks, corals, and other objects of the underwater domain, tend to be and perceived by the state estimation as convex.

Finally, in each planning cycle, the above preprocessing step produces the visual objectives used during planning. Though, not keeping past information of previously detected clusters, could result to a highly sub-optimal reactive behavior. Thus, the set V contains a maximum of m computed visual objectives, in order to ensure real-time planning, and to avoid excessive computation from an ever increasing number of visual objectives. Initially, the visual objectives are added to the list until $|V| = m$. Then, any new measurement replaces the closest one if they are in close proximity by updating the center of the cluster, or in any other case, the oldest one to favor locality and computational efficiency.

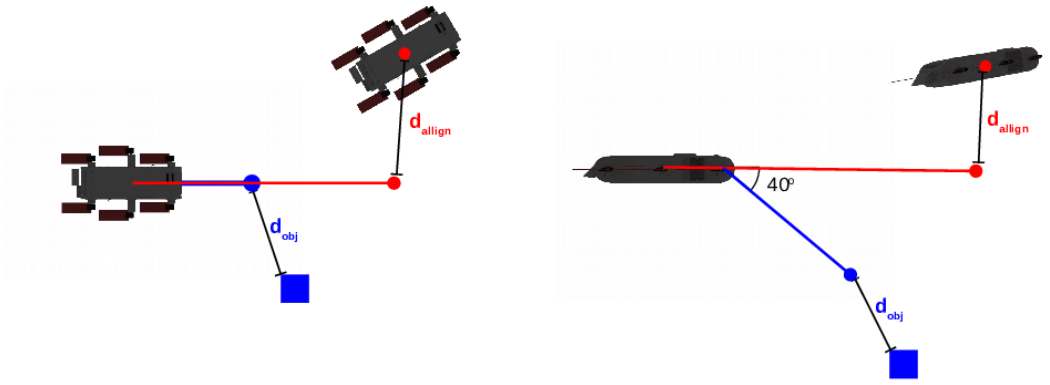
5.4.2 Enhancing Motion Planning

The planning pipeline of AquaNav [17] consists of two modules: First Trajopt [5], which is the main core motion planner producing and refining the paths to be executed, and an assisting warm-restarting process, for supplying paths to Trajopt in the case of failure using BiT-RRT [140]. AquaVis, the proposed framework for adding visibility objectives on AquaNav, modifies only the path optimization problem of Trajopt.

Given the formulation describe previously at Equations 4.8 and 4.10, there are only two additional cost functions that need to be added. The first one greedily forces the robot to view visual objectives, while the second one forces the path to self-correct and maintain the kinematics assumed by the path follower. As expected, these constraints are intuitively opposing, since the second one forces the path to remain valid countering the degrading greedy behavior of the first one. Both functions are described bellow and an example that outlines these novel cost functions is shown in Figure 5.6.

Visibility Constraints

The visibility constraint is the major proposed addition to the AquaNav pipeline, with the sole goal of greedily drive the robot to observe a known set of visual objectives. Defining and utilizing directly in real-time applications the true geometric polytope of the visibility



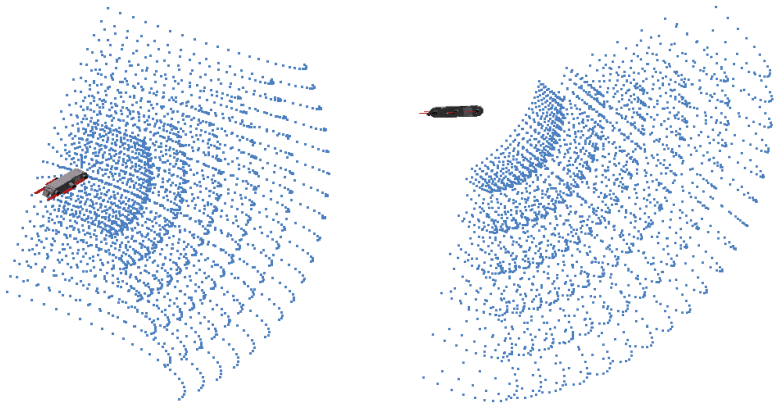
(a) (b)

Figure 5.6 The top (a) and side view (b) of a state using the novel constraints during optimization. The blue square indicates an objective, while the red circle the next waypoint. Minimizing d_{obj} will result on the robot observing the objective, while minimizing d_{align} will result on the robot to be consistent with the kinematics assumed during path execution and planning.

manifold F_s could be very challenging, thus an approximation $F_s^{\sim}$ is used, formed by a representative collection of points. Ideally, these projected points will sufficiently cover the area of the visibility manifold up to a desired resolution. An option is to project points uniformly in different directions and distances from each sensor $c \in C$.

More formally, let T_w^s denoting the transformation of the pose $s \in S$ of the robot in the world coordinate frame, T_r^c the transformation of the pose of camera $c \in C$ in the robot's coordinate frame, K a set of k desired directions, $T_c^k = \begin{bmatrix} R_k & 0_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}$ the transformation describing the k direction of the projected point, and a set D_{vis} of d_{vis} desired distances the robot has to observe a visual objective. Then, assuming a homogeneous multi-sensor configuration, the approximated visibility manifold could be formed as:

$$F_s^{\sim} = \bigcup_{\substack{c\in C \\ k\in K \\ d_{\text{vis}}\in D_{\text{vis}}}} \left\{ T_w^s T_r^c T_c^k \begin{bmatrix} d_{\text{vis}} \\ 0 \\ 0 \\ 1 \end{bmatrix} \right\} \tag{5.4}$$



(a) (b)

Figure 5.7 Different perspectives of the projected points of the $F_s^{\sim}$ visibility set approximating the F_s visibility manifold corresponding to the front camera.

Figure 5.7 depict an example of an approximation manifold for the front camera of Aqua2. For heterogeneous systems with sensors of different field of view and characteristics, in a similar way, user-defined points could be picked off-line to represent the visibility manifold.

Given the above formulation, in order to enable the robot to observe certain visual objectives, the core idea is to minimize the distance between the approximated visibility manifold of the robot at state s with the position of the closest point of interest. Thus, the novel cost function achieving this objective is:

$$\text{Vis(s)} = w' \min_{v \in V} \min_{f \in F_s^{\sim}} ||f - v|| \tag{5.5}$$

Upon successful convergence of the Trajopt optimization process, by definition, at least one visual objective will be visible by at least one sensor in each state.

The parameter w' is a weight that is utilized to adjust the Trajopt optimization in the first iterations: Trajopt uses an adjustable loss factor for collision avoidance to treat it as hard constraint, and after a safe distance from the obstacles has been achieved then it is treated as soft constraint. The idea is to dominate with this weight the optimization in the first steps in order to steer the trajectory in to areas that have more visual objectives, by

using a high value for w' and then to decrease that value so that the original Trajopt cost-functions will be utilized. Thus, it becomes more likely to produce a path that satisfies our goal objective.

This formulation offers many options for developing different policies in the future, such as enforcing visibility of visual objectives from multiple sensors by using an $F_s^{\sim}$ representing the intersection of the FOVs of the desired sensors, maximizing visibility of many objectives by attempting to minimize the sum of the distances between each visual objective and its closest point of $F_s^{\sim}$, and alternate sensing of the target object between different sensors (such as cameras, LIDARs, or sonars) with respect to proximity by choosing the relevant parts of $F_s^{\sim}$ for each state during optimization.

Though the above formulation is powerful and could lead to superior performance in terms of path quality and visibility, at the same time it can add a severe computational overhead during planning, degrading the desired real-time performance of the AquaVis pipeline. However, both desired behavior and real-time replanning can be achieved at the same time, by significantly reducing the approximation quality of $F_s^{\sim}$, to only a single point projected at the center and at a desired distance d_{vis} of each sensor. This might seem reductive but it could be argued that such a trade-off is necessary due to the very limited computational resources of Aqua2, shared by the notoriously computationally heavy SLAM modules. Our experimental results showcase the robustness of the method despite this reduction.

More formally, if d_{vis} is the desired distance for observing visual objectives from each sensor, then the redacted approximated visibility manifold can be constructed as:

$$F_s^{\sim} = \bigcup_{c \in C} \left\{ T_w^s T_r^c \left[d_{\text{vis}} \ 0 \ 0 \ 1 \right]^T \right\} \tag{5.6}$$

The above simple formulation in conjunction with Equation 5.5 guarantees that multiple visual objectives could be observed by multiple sensors along the path, and that at least one objective will be observed at each time if possible. Figure 5.6 shows an instance of the proposed cost function and the distance d_{obj}, shown with blue.

The cost fuction of Equation 5.5 is greedy in the typical flavor of path optimization, since it attempts to steer the robot towards the closest available point of interest for each state. Also, the function is, arguably, rather myopic without providing an easy way to plan paths that are able to observe all given visual objectives, to guarantee that the maximum possible points of interest are observed, or to ensure that at least one visual objective will be observed at each time. However, the underlined intuition is that local optimality, could be extended to global high-quality performance, the same way it is achieved in AquaNav.

On the other hand, such a function could guarantee that during the planned trajectory, multiple objectives could be observed, contrary to the works discussed previously. More importantly, the visual objectives are tracked from a desired proximity, something crucial in the challenging underwater domain, with the very limited visibility range, due to turbidity. Moreover, considering the primary objective on improving vision-based odometry by keeping objects with many features on the frame, most commonly only a subset of visual objectives needs to be tracked. Furthermore, although not studied in this thesis, by developing policies for deciding beforehand which points of interest should be tracked in each step, many local-minimum cases could be avoided, since the entire trajectory is considered during optimization. Specifically, planning with more than one visual objective in the horizon, is a more robust and efficient way to plan trajectories to a desired goal. Especially, if we consider given the short visibility range of the underwater domain combined with a forward moving robot employed with a forward looking camera of the robot.

Finally and most importantly, the weights of the new function introduced could be tuned easily. In other methods, presented above, where there is a mixture of translational and rotational cost functions, tuning the weights is often be very challenging. Especially regarding visibility constraints, cost functions for rotational and translational motions might often lead to conflict and scaling different units can be very difficult (meters vs radians). The proposed cost function offers a way to unite the problem for both types of motion by producing costs with physical and intuitive correspondence. All the costs are measured in

a single metric (meters), assisting tuning. Due to this attribute, the optimization method is allowed to naturally determine the motion, by avoiding the conflicts discussed above, that lead to high non-linearity and non-convexity.

KINEMATIC CONSTRAINTS

The formulation described in the previous section solves the active perception problem sufficiently for holonomic robots, capable of moving freely in $SE(3)$ with no explicit kinematic limitations. Such robots will always orient themselves appropriately and move towards the most convenient visual objectives. Aqua2, though, and the majority of the mobile robots, are non-holonomic systems.

In the case of Aqua2, the controller allows only for 3D Dubins' locomotion, and the path follower inherited by AquaNav performs a simple waypoint navigation. The path follower accepts the 3D coordinates that need to be reached by the robot, along with a constant desired roll orientation during the motion. The robot, ideally is assumed to move on a straight line connecting the previous and the next local goal as described in Chapter 4. Thus, the robot after achieving the waypoint p_{s_i} should maintain an orientation pointing directly to the next waypoint $p_{s_{i+1}}$. In other words, the kinematic constraints assumed during planning is that the robot will always face and move towards the next waypoint. If such constraint is satisfied for every state, then the path produced by the optimization process, will be readily executable by the robot.

This was not an issue that had to be addressed for AquaNav, since the optimization process was focusing on minimizing the path length by satisfying clearance guarantees, which also minimizes rotations. At the same time obstacle avoidance had no significant effect violating the kinematic assumption. By adding the visibility constraint, the robot will be encouraged to face directly the visual objectives, diverging significantly from the desired orientation towards the next waypoint.

The above important issue is resolved by adding another constraint, similar in nature, applied to each state that allows the robot to observe the visual objectives while at the same time it forces the robot to face towards the next waypoint. Such alignment was achieved by projecting a single point in front of the robot at a specific distance and then minimizing the distance d_{align} of this point with the next waypoint. Fig. 5.6 shows an instance of such distance with red.

More formally, let p_s denote the 3D position coordinates of state s at a common fixed coordinate system, $S = [s_1, s_2, \ldots, s_{n-1}, s_n]$ be the trajectory to be optimized, where s_i, $s_{i+1} \in S$ are two consecutive states, that correspond to the two consecutive waypoints p_{s_i} and $p_{s_{i+1}}$. Finally, let $len(S)$ return the total length of the S path, and the average distance between two consecutive states be computed as:

$$av(S) = \frac{len(S)}{n-1} \tag{5.7}$$

Then, if ε is a positive value, for each state s_i the kinematic constraint aligning properly the robot to produce valid trajectories is given by:

$$A(s_i) = w'' \left|\left| T_w^{s_i} \begin{bmatrix} av(S)-\varepsilon \\ 0 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} p_{s_{i+1}}^T \\ 1 \end{bmatrix} \right|\right|, \tag{5.8}$$

Similar to Equation 5.6 the first element of the first vector is the distance that the point will be projected, and w'' a weight of the cost function for the objective function. The distance needs to be automatically adjusted during optimization according to the continuously changing path length, while at the same time it both maintains waypoints of equal distance and encourages minimal paths by being reduced by a small positive value ε.

It is worth noting that similarly to Equation 5.5, only euclidean distances are considered in a single intuitive function, so no special treatment on adjusting the weights between translational and rotational cost functions is needed, and weight tuning is intuitive and simple. By adding Equation 5.5 and Equation 5.8 as cost functions in the optimization formulation of AquaNav, a robust active perception behavior emerges as shown in the next section.

Finally, indeed AquaNav, thus also AquaVis, are pipelines that aim to safely navigate a robot with very challenging and not well-studied dynamics. In order to achieve that, both pipelines assume that the robot operates within some motion uncertainty, which is mitigated with a proper safety clearance. Disturbances during motion are expected also for AquaVis, and the probability of losing track of the desired visual objective exists, especially around the waypoints when switching local goals and desired orientation. Simulation experiments presented in the next section, show that such issues are mitigated with constant replanning since less aggressive motions are expected. Additional object-tracking failure cases due to aggressive motions could be resolved by improving smoothness on the output of the controller, which is not the primary objective of this dissertation.

5.5 Evaluation

Simulation experiments were conducted to validate the robustness of the proposed AquaVis pipeline within the Gazebo simulation environment [106]. To test the effectiveness and robustness of AquaVis to handle arbitrary sensor configurations, two different configurations were used utilizing the single forward looking stereo camera, and the forward looking camera with the bottom looking one.

To simulate the output that is expected to be produced by vision-based SLAM techniques, two simulated LIDARs with the same FOV as the cameras' configuration on the real robot, as shown in Fig. 5.4. The front stereo camera has a horizontal FOV of 120°, a vertical of 90°, and they are tilted downwards by 40°, while the back camera has the exact same FOV but it is tilted downwards by 90°. The LIDARs output depth images of resolution 100×75 which are potentially thousands more than the expected during real deployment, to show the capabilities of the pipeline to deal with large inputs online. In all the experiments, the LIDARs had limited range to simulate the expected turbidity in the real underwater environments.

To extract visual objectives automatically using DBSCAN [185], the maximum distance between features was set to 0.2 m with a minimum number of 5 features per cluster. A maximum set of 15 visual objectives was maintained, with new visual objectives replacing the closest of the old ones that were in a distance less than 0.5 m, or the oldest in the set.

5.5.1 Single Forward Looking Camera

For the first configuration utilizing a single forward-looking camera, the maximum range of the sensor was set to 6 m. AquaVis was tested online against AquaNav in 2 different environments, the Boxes, and the Shipwreck shown in Figure 5.8.

The Boxes Environment, shown in Figure 5.8, is intended to test AquaVis in an environment where feature-rich areas are distributed sparsely in the environment, which is a common scenario in open-water conditions, to the author's experience. AquaNav, by optimizing path length, moves on a straight line, disregarding the features, which are essential for localization. AquaViz, in contrast, reaches the same goal while passing in proximity and observing the feature rich areas (red cubes). The plot in Figure 5.8(e) confirms the effectiveness of the proposed approach to the primary objective as defined in Equation 5.3: AquaNav cannot observe any features for the majority of the time, whereas AquaVis consistently tracked enough features. It is worth noting, that AquaVis introduced a 90° roll to bring the visual objectives of the boxes in the field of view. Moreover, AquaVis maintained tracking for the first 75% of the trajectory that visual objectives could be observed, and lost track at the last 25% where no visual objectives were present that could be observed with a forward looking camera, while the robot moves towards the goal.

Similarly, in the Shipwreck environment, also shown in Figure 5.8, AquaVis was able to observe consistently more features than AquaNav, excluding ascent and descent, which is expected given the kinematics of Aqua2. Also the robot not only oriented itself to track most of the shipwreck but also created the desired proximity, indicating potential use for

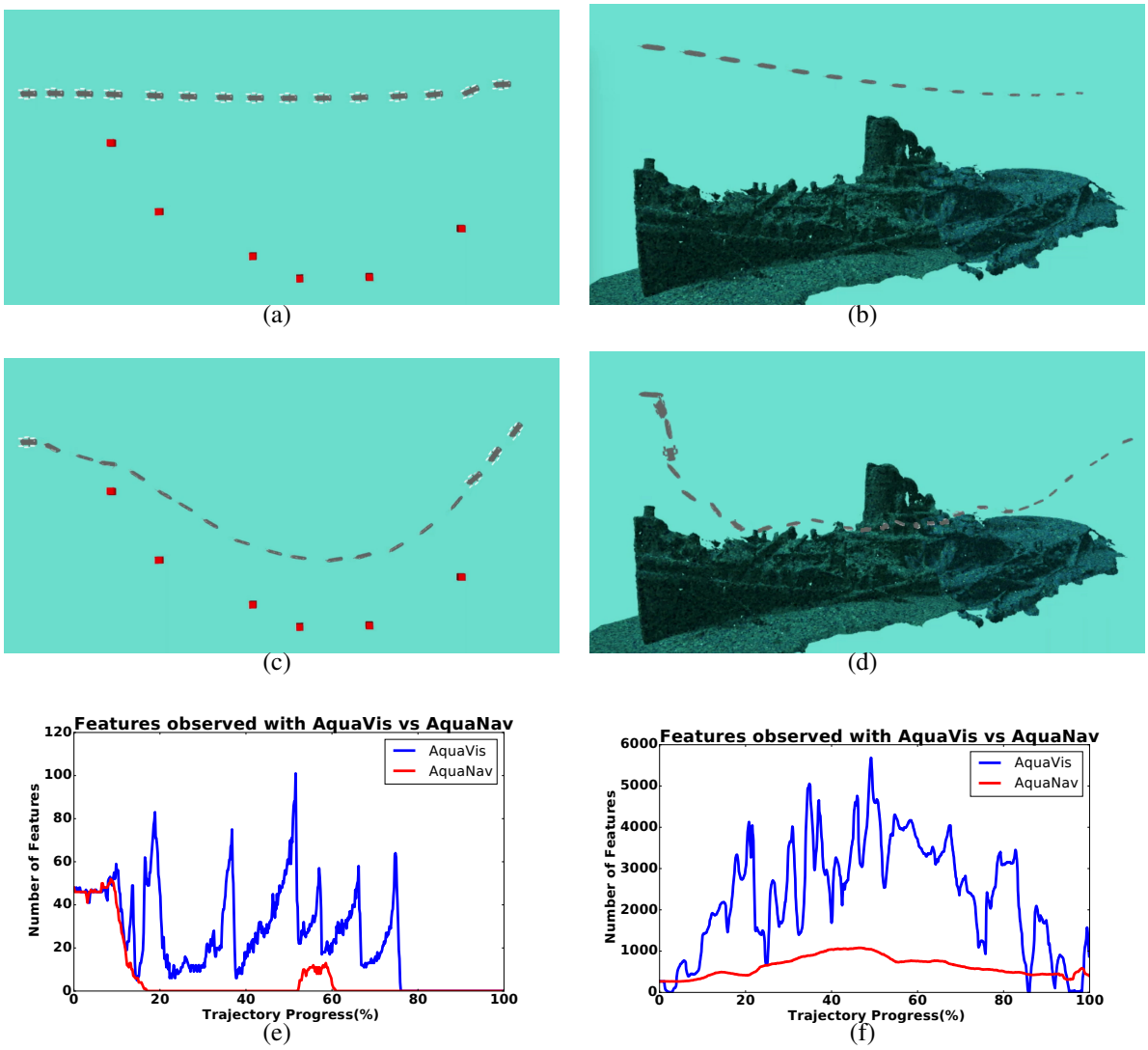


Figure 5.8 The results for the Boxes environment are shown on the left, and for the Shipwreck at the right row. The trajectories produced by AquaNav are shown in (a) and (b), and for AquaVis at (c) and (d). The features observed for both methods are shown in (e) and (f).

mapping purposes. On the other hand, AquaNav moved in a straight line, unaware of the feature rich areas, and tracked only a small portion of the features tracked by AquaVis. This important attribute of AquaVis that indicates strong potential towards mapping challenging unstructured environments from a desired proximity will be shown also in the next section.

5.5.2 HETEROGENEOUS SENSOR SYSTEM

Additional experiments were performed to evaluate the robustness of AquaVis applies on robotic systems with multiple sensors, by utilizing the forward-looking and the bottom-looking camera of the Aqua2 AUV in simulation.

In this section, two different versions of AquaVis are shown: (i) AquaVis-Mono that utilizes only the front camera both for extracting visual objectives and for registering obstacles, and (ii) AquaVis-Dual that additionally utilizes the back camera for informing the objectives extraction method and for sensing. Moreover, the maximum range of the sensors was set to different distances – 3 m for the front cameras and 6 m for the back one – to validate the behavior of AquaVis for an AUV employed with a heterogeneous multi-sensor system with different range capabilities. The desired distance from the visual objectives was set to 1.5 m, half of the maximum range of the front cameras, to encourage observations from the front camera system, and the desired clearance to 0.6 m similarly to the original AquaNav pipeline which produces a safe behavior for the expected operating speed of the robot at $0.4\,\mathrm{m/s}$.

The primary motivation for this work is to enhance underwater operations both for improving state estimation and for enabling mapping, inspection, and monitoring missions. For this purpose AquaVis-Mono and AquaVis-Dual were compared against the original AquaNav framework as baseline in two different environments.

ASSISTING VISION-BASED STATE ESTIMATION

The first environment, the same as the Boxes environment from the previous section, aims to test the capabilities of the three frameworks to produce motions that provide good features and robustify underwater SLAM in environments where they are concentrated in few sparse feature-rich areas; a very common real scenario during underwater deployments. The resulted trajectories are shown in Fig. 5.9 for the AquaNav, the AquaVis-Mono, and the AquaVis-Dual pipelines respectively, and the features tracked by the front camera, and

by both cameras in Fig. 5.10. The robot started from an initial position from which only a small segment of the first box was visible only from the front camera while nothing was visible from the bottom camera, and a goal was set 25 m forward.

Notice that AquaNav focuses on minimizing the path length without regard for tracking features, while AquaVis-Mono was able to track few feature-rich areas with the front camera and all of them with the back. On the other hand, AquaVis-Dual tracked all the obstacles with both cameras, since the back camera feed was informing the path planning and assisting visibility from the front cameras. Such results showcase also the capabilities of AquaVis for heterogeneous sensor systems. For example, the back camera could be considered equivalent to a sonar that has a significantly larger range than a camera underwater. Such sensor configurations could drive the robot towards potentially feature-rich areas detected by sonars from distance, and assist a visual SLAM module using the front stereo cameras.

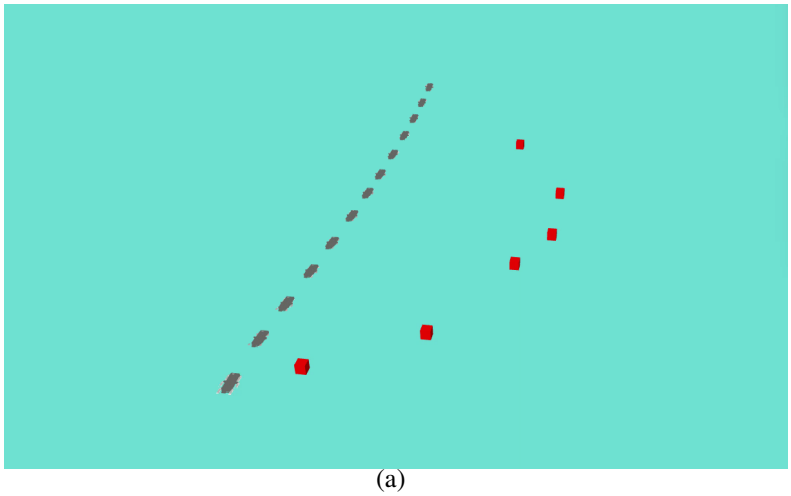
Enabling Mapping and Exploration

The second environment, called the Shipwreck environment, aims to compare the three frameworks on producing motions for mapping, monitoring, and exploring challenging underwater structures, such as shipwrecks; a significant motivation for this study and our previous works. The resulted trajectories are shown in Fig. 5.11 for the AquaNav, the AquaVis-Mono, and the AquaVis-Dual pipelines respectively, and the features tracked by the front camera, and by both cameras in Fig. 5.12. The initial pose of the robot did not allow it to see any features from the front camera, while a small segment of the shipwreck was visible from the back camera. The goal was set at 45 m forward, the approximate length of the shipwreck.

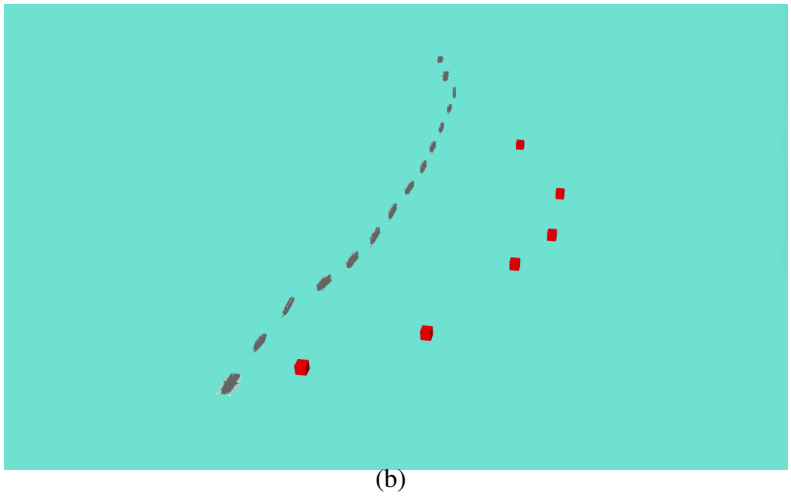
AquaNav and AquaVis-Mono had very similar performance, hardly observing any features with the front cameras, while only a small segment of the deck was captured by the back camera, due to its long sensing range. On the other hand, AquaVis-Dual was capa-

ble to observe a significant segment of the shipwreck with the short-sighted front cameras, while the majority of the shipwreck with both cameras. Moreover, using AquaVis-Dual the Aqua2 was driven in very close proximity to the shipwreck, which showcases the inherited strong safety and obstacle avoidance guarantees.

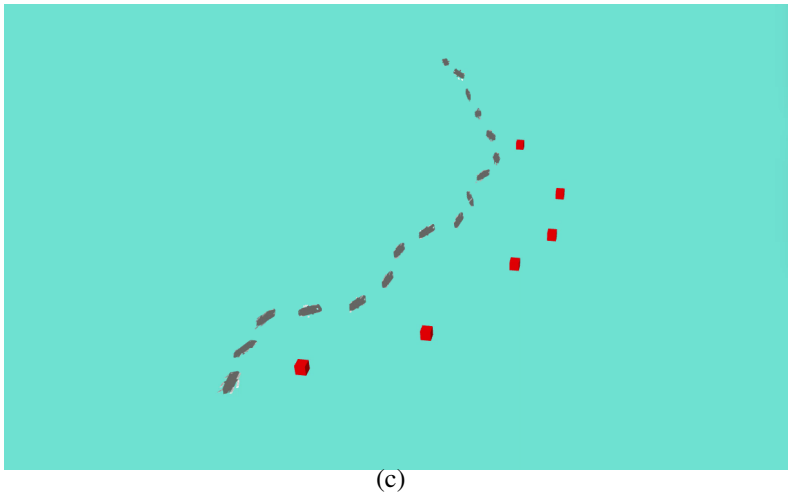
The results show great potential for application on robotic platforms with heterogeneous multi-sensor configurations, such as sonars with cameras. The pipeline allows replacing effortlessly the back camera, or even appending the current configuration, with a sonar on the Aqua2, to automatically detect with low resolution potential points of interest. In such configuration, AquaVis could drive the robot towards points of interest detected from distance using sonars to take close visual observations; a very useful attribute for many scientific, commercial, and security underwater operations.



(a)



(b)



(c)

Figure 5.9 Trajectories produced by (a) AquaNav, (b) AquaVis-Mono, and (c) AquaVis-Dual for the Boxes environment.

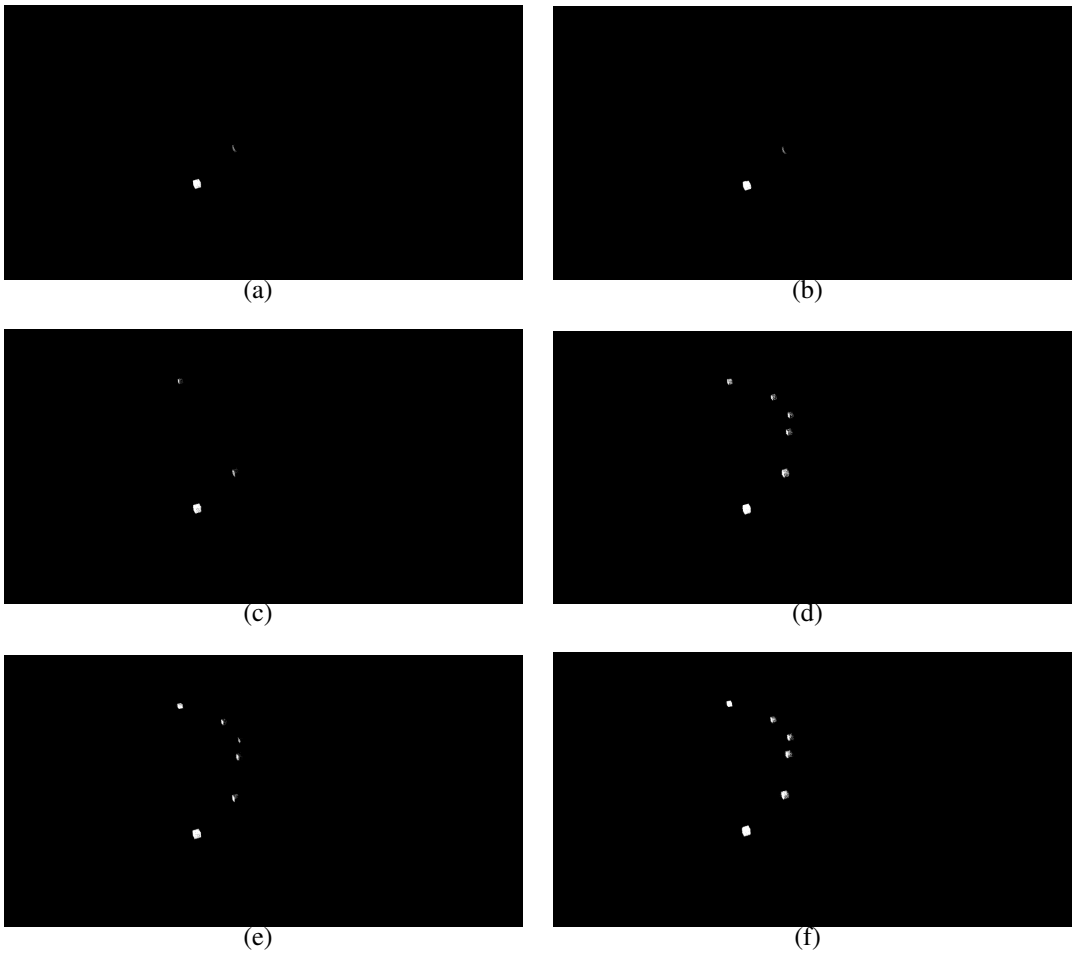


Figure 5.10 First column: The corresponding point cloud obtained by the front camera for the Boxes environment with (a) AquaNav, (c) AquaVis-Mono, and (e) AquaVis-Dual . Second column: The corresponding point cloud obtained by both cameras with (b) AquaNav, (d) AquaVis-Mono, and (f) AquaVis-Dual. As expected, adding a second back camera of larger sensing range, informs planning, robustifying the behavior further, while the limited range of the front cameras for AquaVis, leads to similar behavior with the uninformed AquaNav pipeline.

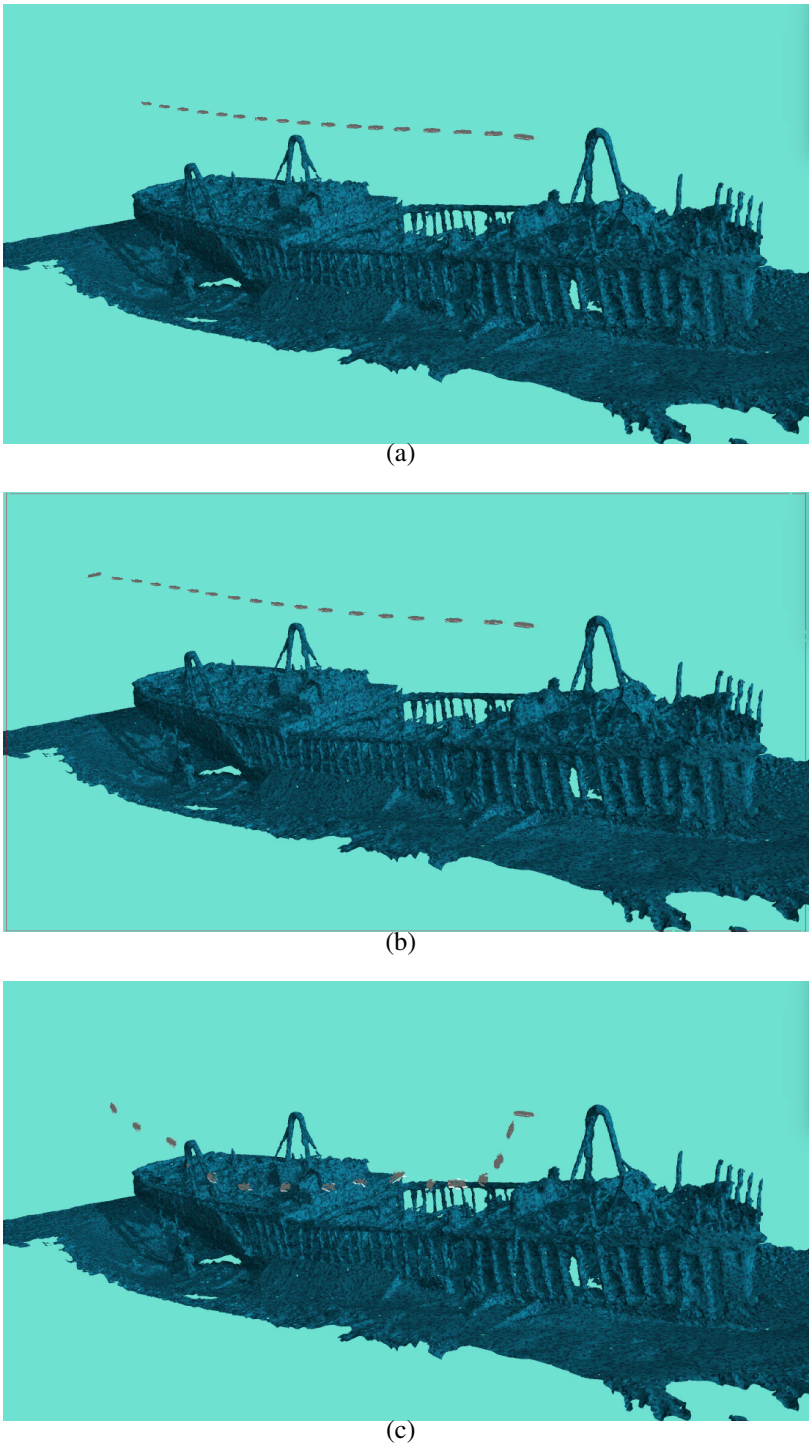


Figure 5.11 Trajectories produced by (a) AquaNav, (b) AquaVis-Mono, and (c) AquaVis-Dual for the Shipwreck environment.

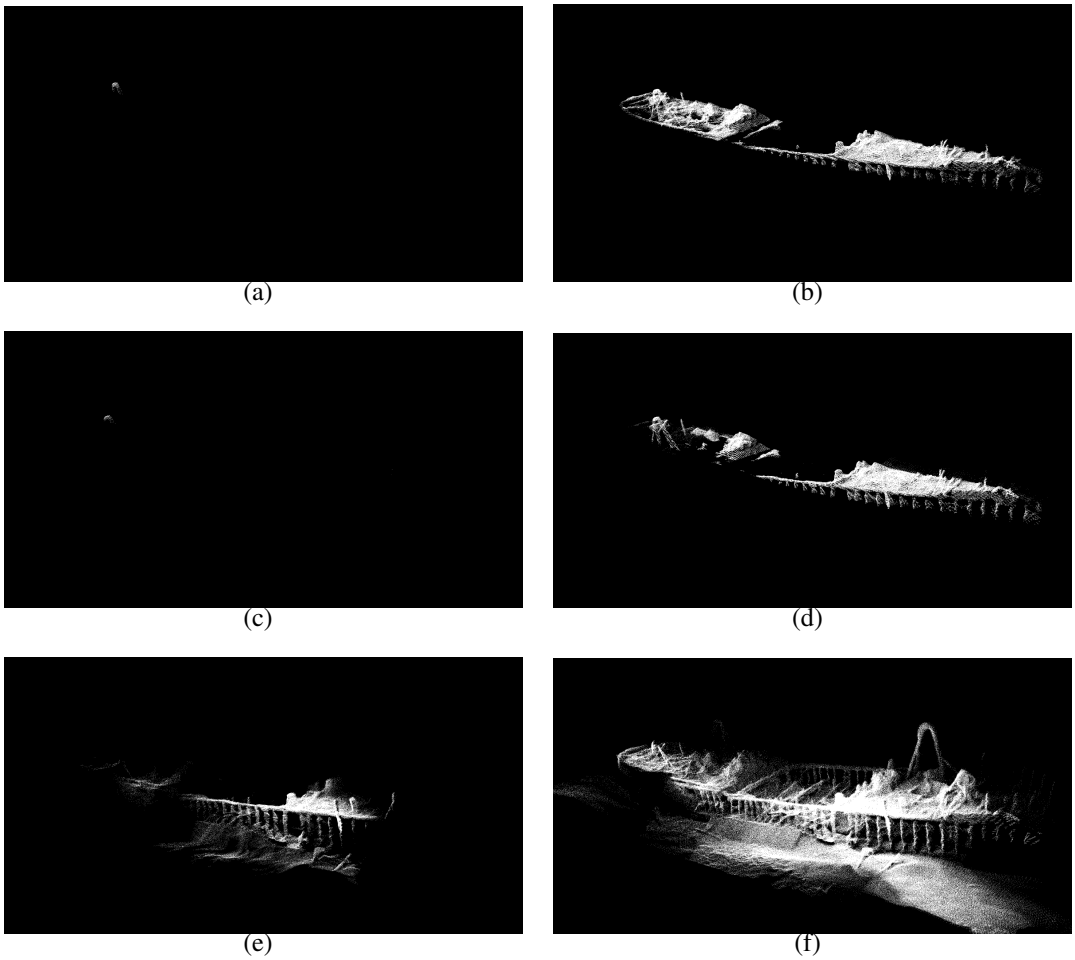


Figure 5.12 First column: The corresponding point cloud obtained by the front camera for the Shipwreck environment with (a) AquaNav, (c) AquaVis-Mono, and (e) AquaVis-Dual . Second column: The corresponding point cloud obtained by both cameras with (b) AquaNav, (d) AquaVis-Mono, and (f) AquaVis-Dual. As expected, adding a second back camera of larger sensing range, informs planning, robustifying the behavior further, while the limited range of the front cameras for AquaVis, leads to similar behavior with the uninformed AquaNav pipeline.

CHAPTER 6

DISCUSSION AND FUTURE DIRECTIONS

In this chapter, potential future applications and theoretical contributions using the proposed research are discussed.

6.1 RRT$^+$

The core idea of RRT$^+$ has been already utilized and extended beyond the limits of Chapter 3 by other researchers in the community [186], who and provided alternative solutions for defining subspaces and determining timeouts. More importantly, they extended the experimental validation towards more challenging realistic scenarios. Similarly to that work, there are four future directions for the RRT$^+$ family of planners: Utilizing the idea on new planners, developing new ways for defining relevant subspaces, determining more effective timeout limits for each subsearch, further showcasing the robustness of the technique with new applications.

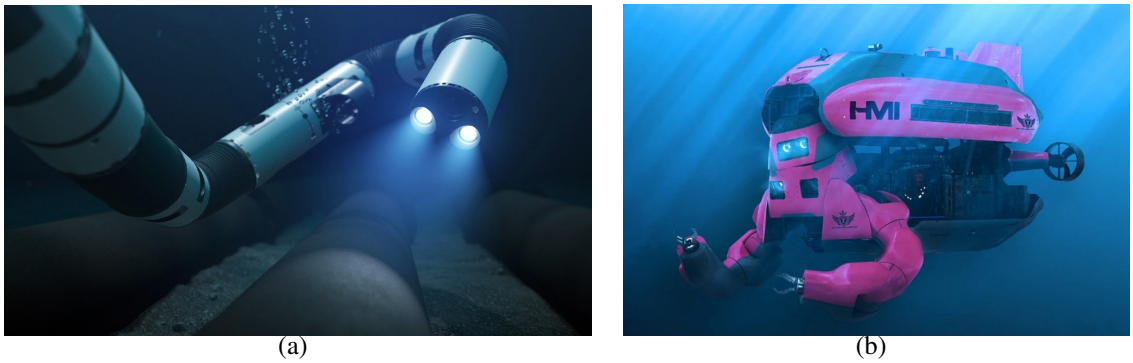
One of the important characteristics of RRT$^+$ is that it is a very general concept that can be applied to many different sampling-based planners, even beyond the RRT-based ones. For example, the same sampling technique could be applied to KPIECE [61] to exploit the biasing of sampling to unexplored areas by projecting these samples to the lower dimensional subspaces. A more counter-intuitive direction will be to identify the fundamental disadvantages of progressively searching for a solution in lower-dimensional subspaces and mitigate them with robust planners designed to overcome such issues. For instance, lower-dimension subspaces — if they contain a solution — generally suffer from narrow passages. Integrating the RRT$^+$ subspace sampling with STRIDE [6], a planner

that performs best in high-dimensional spaces with narrow passages, seem a very natural and promising idea. RRT+ could also be integrated in a higher-level within a multimodal motion planning framework for generating quickly paths, utilized or processed by other planners, such as sampling-based near-optimal, or path-optimization planners.

A major concept and problem introduced, was defining lower-dimensional subspaces where a solution possibly exists. To showcase the robustness of the technique even with naive and simple approaches, where only linear constraints were considered, the subspaces were formed only be a set of independent DoFs and a group mutually constrained ones, and the subspaces were random. There are several improvements that future research could accomplish. Firstly, instead of using only one set of constrained and one of set of fully independent DOFs, subspaces could be formed by adding different linear constraints to different DoFs. Such sets could either include DoFs that are unlikely to be important for finding a solution, or by utilizing metrics such as the distance between the initial and the goal configuration. Secondly, linear constraints could be replaced by non-linear ones that represent projections of the free Configuration Space better. Deep learning techniques could be employed to learn such subspaces from past queries, while they could also be used to prioritize search into subspaces that show strong likelihood of finding a solution fast for a specific query.

Regarding effectively determining the termination condition for a subsearch, a better solution might be to use lower-dimensional projections to monitor the coverage of the subspace, similarly to KPIECE [61], and switch to a different subspace if the progression of the tree is stalled; an indication of disconnected free configuration subspace or very narrow passages. Alternatively, other metrics could be developed for detecting these cases, such as density-based conditions by monitoring the average distance of new samples to existing nodes, or greedy ones by measuring the progression of the tree towards the goal.

Finally, as robots become inevitably more complex, with more actuators added, thus more DoFs, RRT$^+$ and other approaches dealing with high-dimensionality are expected to



(a) (b)

Figure 6.1 Examples of underwater robotic platforms that AquaNav and AquaVis could be applied: (a) The Eelume© underwater snake robot, and (b) the Aquanaut AUV by HMI.

become more relevant in real applications. The author aspires to apply such techniques to the systems this thesis is focusing on: complex mobile manipulators, humanoids, and multirobot systems, moving safely and robustly in dynamic environments on land, in the water, in the sky, or beyond.

6.2 AquaNav and AquaVis

Regarding AquaNav and AquaVis, future research could focus on both major improvements in the pipeline and new extensions, and also real applications and testing.

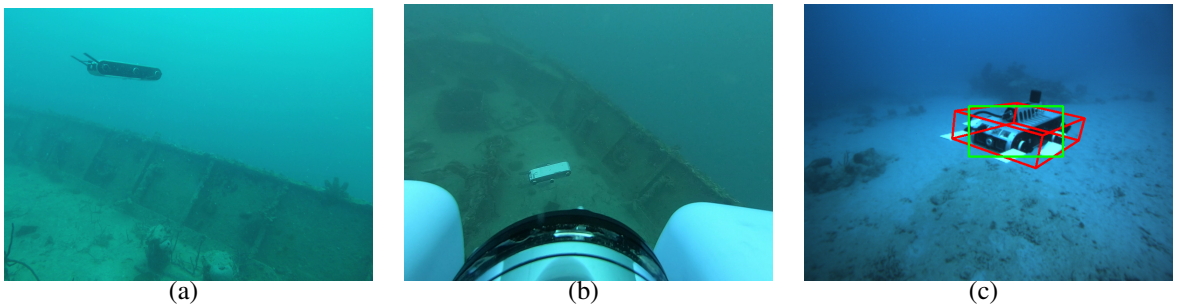
Currently, the planning module assumes a known constant speed and a fixed clearance for every state. Modifications to the core optimization process, could allow planning with dynamically defined clearance at each step, in compliance with the kinematics of the robot, to improve safety and performance. Such modifications applied on a robot with known dynamics, could be paired with a model predictive control [187] scheme — that requires known boundaries for the optimization — for superior behavior and performance. Another interesting problem, will be to plan with dynamic obstacles with strong clearance guarantees. Such research direction would involve tracking, behavior prediction, and modifying

the optimization process. Improving the replanning time by accelerating path planning and point-cloud processing with GPUs is also a very promising area of research.

Although the primary focus of both AquaNav and AquaVis was application in the underwater domain, both are very general frameworks with no added assumption that are true only in the underwater domain. Thus, it would be interesting to utilize these pipelines for mobile robots operating in indoor or outdoor environments, in the air or in the space. Especially, both frameworks, from a design choice, are able to operate on arbitrary complex robotic systems, such as kinematic chains, or robots employed with manipulators, Figure 6.1. A very interesting research direction for AquaVis will be to utilize it on mobile robots with multiple sensors mounted on multiple manipulators, to improve real-time tracking, mapping, and monitoring. AquaVis with the current formulation attempts to observe at least one visual objective at each state. A new formulation and a higher level planner could utilize AquaVis to maximize coverage of a certain area of interest or for performing active exploration of unknown environments.

Unfortunately, due to unavailability of the Aqua2 platform due to the pandemic of COVID-19 and supply chain issues in electronics, only few open-water trials were performed for AquaNav, while AquaVis was validated purely in simulation. Future work will focus on testing AquaNav and AquaVis in challenging dynamic environments, such as underwater caves, shipwrecks, fishfarms, and for mapping, monitoring and inspecting other natural structures and human infrastructure. Towards that research direction, AquaVis is expected to be utilized for exploration, mapping and tracking as part of a large multi-institutional and multi-PI project (NSF-2024741) that aims to map shipwrecks with a multi-robot scheme [188].

The main idea of the multi-robot wreck exploration approach is to have a team of co-robots collaborating with a human operator. There are two types of robots: proximal observers, which will operate close to the structure in order to produce an accurate map, and distal observers, which will be at distance maintaining the global picture of the structure



(a) (b) (c)

Figure 6.2 Key components of the proposed multi-robot shipwreck exploration approach: (a) the proximal observer navigating and observing in proximity the shipwreck, (b) the distal observer tracking the proximal observer and maintaining a wider view of the general structure of the shipwreck, and (c) an instance of the proposed approach for the distal observer to track the proximal observer introduced in previous work [9].

and the pose of the proximal observer, Figure 6.2. Critical components of this application is the motion of the proximal observer to map the shipwreck, a highly unstructured cluttered environment, and the motion of the distal observer to keep the shipwreck and the proximal observer into frame. Very promising experimental results from AquaVis have indicated that AquaVis paired with a high level exploration planner can effectively explore a target structure with safety from a desired proximity. The high-level planner will assign goal positions for the robot to achieve, and visual objectives to perceive encouraging observations from unknown relevant areas. On the other hand, AquaVis could be also applied to the distal observer to guarantee that the proximal observer and the distal observer remain in the desired orientation and distance. Similar to the proximal observer, a high-level planner taking into account the turbidity and the expected future position of the proximal observer, could inform AquaVis to drive the distal observer in a safe and efficient way.

CHAPTER 7

CONCLUSION

This dissertation provided real-time computational methods solving important problems in emerging technologies such as high-dimensional motion planning, 3D underwater navigation, and 3D active perception, along with extensive validation of these techniques in simulation and real-world experiments. These contributions offer fundamental tools and ideas that enable robots to solve very challenging motion planning and underwater navigation problems in real-time for the first time, while simultaneously offering a great potential for further improvements and enhancements.

Each method is general enough to be applicable to a wide variety of complex robotic systems that need to make fast decisions on how to move safely in order to avoid obstacles and observe areas of interest. Thus, not only were new problems of significant complexity solved in real-time by the robotic platforms presented in this thesis, but the proposed methodologies can be applied to other platforms operating in different domains beyond underwater, from indoor and outdoor operations, to aerial and space missions.

The author aspires to ensure that the key ideas and formulation introduced in this dissertation will inspire and support future research towards (a) advancing humanity's collective well-being by eliminating dangerous, repetitive, and emotionally unfulfilling manual labor; (b) improving human life conditions by utilizing the efficiency and social surplus of robotic research in construction, manufacturing, health and elderly care; and (c) expanding our knowledge by developing robust robotic systems that can explore and operate in domains far beyond the reach of our biology.

BIBLIOGRAPHY

[1] Oussama Khatib, Xiyang Yeh, Gerald Brantner, Brian Soe, Boyeon Kim, Shameek Ganguly, Hannah Stuart, Shiquan Wang, Mark Cutkosky, Aaron Edsinger, et al. Ocean one: A robotic avatar for oceanic discovery. *IEEE Robotics & Automation Magazine*, 23(4):20–29, 2016.

[2] James J Kuffner and Steven M LaValle. RRT-connect: An efficient approach to single-query path planning. In *IEEE Int. Conf. on Robotics and Automation*, volume 2, pages 995–1001, 2000.

[3] Sertac Karaman and Emilio Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems VI*, 104(2), 2010.

[4] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *2009 IEEE International Conference on Robotics and Automation*, pages 489–494. IEEE, 2009.

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

[6] Bryant Gipson, Mark Moll, and Lydia E Kavraki. Resolution independent density estimation for motion planning in high-dimensional spaces. In *IEEE Int. Conf. on Robotics and Automation*, pages 2437–2443, 2013.

[7] John Schulman, Jonathan Ho, Alex X Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Robotics: science and systems*, volume 9, pages 1–10. Citeseer, 2013.

[8] Sharmin Rahman, Alberto Quattrini Li, and Ioannis Rekleitis. Sonar Visual Inertial SLAM of Underwater Structures. In *IEEE International Conference on Robotics and Automation*, pages 5190–5196, Brisbane, Australia, May 2018.

[9] Bharat Joshi, Md Modasshir, Travis Manderson, Hunter Damron, Marios Xanthidis, Alberto Quattrini Li, Ioannis Rekleitis, and Gregory Dudek. Deepurl: Deep pose estimation framework for underwater relative localization. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1777–1784, 2020.

[10] Martin Buehler, Daniel E Koditschek, and PJ Kindlmann. A simple juggling robot: Theory and experimentation. In *Experimental Robotics I*, pages 35–73. Springer, 1990.

[11] Jason M O'Kane and Steven M LaValle. Localization with limited sensing. *IEEE Transactions on Robotics*, 23(4):704–716, 2007.

[12] Jeremy S Lewis and Jason M O'Kane. Guaranteed navigation with an unreliable blind robot. In *2010 IEEE International Conference on Robotics and Automation*, pages 5519–5524. IEEE, 2010.

[13] Jeremy S Lewis and Jason M O'Kane. Reliable indoor navigation with an unreliable robot: Allowing temporary uncertainty for maximum mobility. In *2012 IEEE International Conference on Robotics and Automation*, pages 160–165. IEEE, 2012.

[14] Jeremy S Lewis, Daniel A Feshbach, and Jason M O'Kane. Guaranteed coverage with a blind unreliable robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7383–7390. IEEE, 2018.

[15] Arthur D Kuo. A mechanical analysis of force distribution between redundant, multiple degree-of-freedom actuators in the human: Implications for the central nervous system. *Human movement science*, 13(5):635–663, 1994.

[16] Marios Xanthidis, Joel M Esposito, Ioannis Rekleitis, and Jason M O'Kane. Motion planning by sampling in subspaces of progressively increasing dimension. *Journal of Intelligent & Robotic Systems*, pages 1–13, 2020.

[17] Marios Xanthidis, Nare Karapetyan, Hunter Damron, Sharmin Rahman, James Johnson, Allison O'Connell, Jason M O'Kane, and Ioannis Rekleitis. Navigation in the presence of obstacles for an agile autonomous underwater vehicle. In *IEEE Int. Conf. on Robotics and Automation*, pages 892–899, 2020.

[18] Gregory Dudek, Michael Jenkin, Chriss Prahacs, Andrew Hogue, Junaed Sattar, Philippe Giguere, Andrew German, Hongyu Liu, Shane Saunderson, Arlene Ripsman, Saul Simhon, Luz Abril Torres-Mendez, Evangelos Milios, Pifu Zhang, and Ioannis Rekleitis. A visually guided swimming robot. In *IEEE/RSJ International*

Conference on Intelligent Robots and Systems (IROS), pages 1749–1754, Edmonton AB, Canada, Aug. 2005.

[19] Marios Xanthidis, Michail Kalaitzakis, Nare Karapetyan, James Johnson, Nikolaos Vitzilaios, Jason M. O'Kane, and Ioannis Rekleitis. AquaVis: A perception-aware autonomous navigation framework for underwater vehicles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5387–5394, Prague, Czech Republic, 2021.

[20] Alberto Quattrini Li, Adem Coskun, Sean M. Doherty, Shervin Ghasemlou, Apoorv S. Jagtap, MD Modasshir, Sharmin Rahman, Akanksha Singh, Marios Xanthidis, Jason M. O'Kane, and Ioannis Rekleitis. Experimental comparison of open source vision based state estimation algorithms. In *International Symposium of Experimental Robotics (ISER)*, Tokyo, Japan, Mar. 2016.

[21] Alberto Quattrini Li, Adem Coskun, Sean M. Doherty, Shervin Ghasemlou, Apoorv S. Jagtap, Md Modasshir, Sharmin Rahman, Akanksha Singh, Marios Xanthidis, Jason M. O'Kane, and Ioannis Rekleitis. Vision-based shipwreck mapping: on evaluating features quality and open source state estimation packages. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–10. IEEE, 2016.

[22] Bharat Joshi, Sharmin Rahman, Michail Kalaitzakis, Brennan Cain, James Johnson, Marios Xanthidis, Nare Karapetyan, Alan Hernandez, Alberto Quattrini Li, Nikolaos Vitzilaios, and Ioannis Rekleitis. Experimental Comparison of Open Source Visual-Inertial-Based State Estimation Algorithms in the Underwater Domain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7221–7227, Macau, Nov. 2019.

[23] Bharat Joshi, Marios Xanthidis, Sharmin Rahman, and Ioannis Rekleitis. High definition, inexpensive, underwater mapping. In *2022 IEEE International Conference on Robotics and Automation (ICRA)*, Accepted, 2022. IEEE.

[24] Alberto Quattrini Li, Marios Xanthidis, Jason M O'Kane, and Ioannis Rekleitis. Active localization with dynamic obstacles. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1902–1909. IEEE, 2016.

[25] Marios Xanthidis, Alberto Quattrini Li, and Ioannis Rekleitis. Shallow coral reef surveying by inexpensive drifters. In *OCEANS 2016-Shanghai*, pages 1–9. IEEE, 2016.

[26] Marios Xanthidis, Kostantinos J. Kyriakopoulos, and Ioannis Rekleitis. Dynamically Efficient Kinematics for Hyper-Redundant Manipulators. In *The 24th Mediter-*

ranean Conf. on Control and Automation, pages 207–213, Athens, Greece, Jun. 2016.

[27] Steven M LaValle. *Planning Algorithms*. Cambridge Univ. press, 2006.

[28] John H Reif. Complexity of the mover's problem and generalizations. In *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pages 421–427. IEEE, 1979.

[29] John Canny. *The complexity of robot motion planning*. MIT press, 1988.

[30] Lydia E Kavraki, Petr Švestka, Jean-Claude Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. on Robotics and Automation*, 12(4):566–580, 1996.

[31] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report TR 98-11, Computer Science Dept., Iowa State University, Oct. 1998.

[32] Nathan A Wedge and Michael S Branicky. On heavy-tailed runtimes and restarts in rapidly-exploring random trees. In *Twenty-third AAAI conference on artificial intelligence*, pages 127–133, 2008.

[33] Nancy M Amato and Yan Wu. A randomized roadmap method for path and manipulation planning. In *Proceedings of IEEE international conference on robotics and automation*, volume 1, pages 113–120. IEEE, 1996.

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

[35] Valérie Boor, Mark H Overmars, and A Frank Van Der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, volume 2, pages 1018–1023. IEEE, 1999.

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

[37] Jerome Barraquand and Jean-Claude Latombe. *On nonholonomic mobile robots and optimal maneuvering*. Stanford University, Center for Integrated Facility Engineering, 1989.

[38] Emmanuel Mazer, Juan Manuel Ahuactzin, and Pierre Bessiere. The ariadne's clew algorithm. *Journal of Artificial Intelligence Research*, 9:295–316, 1998.

[39] David Hsu, Jean-Claude Latombe, and Rajeev Motwani. Path planning in expansive configuration spaces. In *Proceedings of International Conference on Robotics and Automation*, volume 3, pages 2719–2726. IEEE, 1997.

[40] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[41] Hemant M Kakde. Range searching using kd tree. *from the citeseerx database on the World Wide Web: http://citeseerx. ist. psu. edu/viewdoc/summary*, 2005.

[42] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2997–3004. IEEE, 2014.

[43] Lucas Janson, Edward Schmerling, Ashley Clark, and Marco Pavone. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *The International journal of robotics research*, 34(7):883–921, 2015.

[44] Jonathan D Gammell, Siddhartha S Srinivasa, and Timothy D Barfoot. Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3067–3074. IEEE, 2015.

[45] Sanjiban Choudhury, Jonathan D Gammell, Timothy D Barfoot, Siddhartha S Srinivasa, and Sebastian Scherer. Regionally accelerated batch informed trees (rabit*): A framework to integrate local information into optimal path planning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4207–4214. IEEE, 2016.

[46] SS Srinivasa, Timothy D Barfoot, and JD Gammell. Batch informed trees (bit*): Informed asymptotically optimal anytime search. *The International Journal of Robotics Research*, 39(5), 2020.

[47] Marlin Polo Strub and Jonathan D Gammell. Adaptively informed trees (ait): Fast asymptotically optimal path planning through adaptive heuristics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3191–3198. IEEE, 2020.

[48] Marlin Polo Strub and Jonathan D Gammell. Advanced bit (abit*): Sampling-based planning with advanced graph-search techniques. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 131–136. IEEE, 2020.

[49] Lydia E Kavraki and Jean-Claude Latombe. Probabilistic roadmaps for robot path planning. *Pratical motion planning in robotics: current aproaches and future challenges*, pages 33–53, 1998.

[50] Pang C Chen and Yong K Hwang. Sandros: a dynamic graph search algorithm for motion planning. *IEEE Transactions on Robotics and Automation*, 14(3):390–403, 1998.

[51] Roland Geraerts and Mark H Overmars. Creating high-quality roadmaps for motion planning in virtual environments. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4355–4361. IEEE, 2006.

[52] Sean Quinlan and Oussama Khatib. Elastic bands: Connecting path planning and control. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 802–807. IEEE, 1993.

[53] Oliver Brock and Oussama Khatib. Elastic strips: A framework for motion generation in human environments. *The International Journal of Robotics Research*, 21(12):1031–1052, 2002.

[54] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *2011 IEEE international conference on robotics and automation*, pages 4569–4574. IEEE, 2011.

[55] Chonhyon Park, Jia Pan, and Dinesh Manocha. Itomp: Incremental trajectory optimization for real-time replanning in dynamic environments. In *Twenty-Second International Conference on Automated Planning and Scheduling*, 2012.

[56] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[57] Anthony Stentz. Optimal and efficient path planning for partially known environments. In *Intelligent unmanned ground vehicles*, pages 203–220. Springer, 1997.

[58] Sven Koenig and Maxim Likhachev. Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics*, 21(3):354–363, 2005.

[59] Joel M Esposito. Conditional density growth (cdg) model: a simplified model of rrt coverage for kinematic systems. *Robotica*, 31(05):733–746, 2013.

[60] Léonard Jaillet, Juan Cortés, and Thierry Siméon. Transition-based RRT for path planning in continuous cost spaces. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2145–2150, 2008.

[61] Ioan A Şucan and Lydia E Kavraki. Kinodynamic motion planning by interior-exterior cell exploration. In *Algorithmic Foundation of Robotics VIII*, pages 449–464. Springer, 2009.

[62] Ioan A Şucan, Mark Moll, and Lydia E Kavraki. The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, 2012.

[63] Sachin Chitta, Ioan Sucan, and Steve Cousins. Moveit![ros topics]. *IEEE Robotics & Automation Magazine*, 19(1):18–19, 2012.

[64] John H Reif. Complexity of the generalized mover's problem. Technical report, Harvard University, Cambridge, MA Aiken Computation Lab, 1985.

[65] Jung Jun Park, Hwi Su Kim, and Jae-Bok Song. Collision-free path planning for a redundant manipulator based on PRM and potential field methods. *Journal of Institute of Control, Robotics and Systems*, 17(4):362–367, 2011.

[66] Dominik Bertram, James Kuffner, Ruediger Dillmann, and Tamim Asfour. An integrated approach to inverse kinematics and path planning for redundant manipulators. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1874–1879, 2006.

[67] Mike Vande Weghe, Dave Ferguson, and Siddhartha S Srinivasa. Randomized path planning for redundant manipulators without inverse kinematics. In *7th IEEE-RAS Int. Conf. on Humanoid Robots*, pages 477–482, 2007.

[68] Yang Qian and Ahmed Rahmani. Path Planning Approach for Redundant Manipulator Based on Jacobian Pseudoinverse-RRT Algorithm. In *6th Int. Conf. on Intelligent Robotics and Applications*, pages 706–717, Busan, South Korea, Sep. 2013.

[69] John Vannoy and Jing Xiao. Real-time adaptive motion planning (ramp) of mobile manipulators in dynamic environments with unforeseen changes. *IEEE Trans. on Robotics*, 24(5):1199–1212, 2008.

[70] Dmitry Berenson, James Kuffner, and Howie Choset. An optimization approach to planning for mobile manipulation. In *IEEE Int. Conf. on Robotics and Automation*, pages 1187–1192, 2008.

[71] Jur P Van Den Berg and Mark H Overmars. Prioritized motion planning for multiple robots. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 430–435, 2005.

[72] Stefano Carpin and Enrico Pagello. On parallel RRTs for multi-robot systems. In *Proc. 8th Conf. Italian Association for Artificial Intelligence*, pages 834–841, 2002.

[73] Glenn Wagner. Subdimensional expansion: A framework for computationally tractable multirobot path planning. *Master thesis*, 2015.

[74] Takahiro Otani and Makoto Koshino. Applying a path planner based on RRT to cooperative multirobot box-pushing. *Artificial Life and Robotics*, 13(2):418–422, 2009.

[75] Kiril Solovey, Oren Salzman, and Dan Halperin. Finding a needle in an exponential haystack: Discrete rrt for exploration of implicit roadmaps in multi-robot motion planning. In *Algorithmic Foundations of Robotics XI*, pages 591–607. Springer, 2015.

[76] James J Kuffner Jr, Satoshi Kagami, Koichi Nishiwaki, Masayuki Inaba, and Hirochika Inoue. Dynamically-stable motion planning for humanoid robots. *Autonomous Robots*, 12(1):105–118, 2002.

[77] James Kuffner, Koichi Nishiwaki, Satoshi Kagami, Masayuki Inaba, and Hirochika Inoue. Motion planning for humanoid robots. In *The Eleventh Int. Symposium Robotics Research*, pages 365–374, 2005.

[78] Hong Liu, Qing Sun, and Tianwei Zhang. Hierarchical RRT for humanoid robot footstep planning with multiple constraints in complex environments. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3187–3194, 2012.

[79] Paul Vernaza and Daniel D Lee. Efficient dynamic programming for high-dimensional, optimal motion planning by spectral learning of approximate value function symmetries. In *IEEE Int. Conf. on Robotics and Automation*, pages 6121–6127, 2011.

[80] Eiichi Yoshida. Humanoid motion planning using multi-level dof exploitation based on randomized method. In *2005 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3378–3383, 2005.

[81] Andrew Wells and Erion Plaku. Adaptive sampling-based motion planning for mobile robots with differential constraints. In *Conference Towards Autonomous Robotic Systems*, pages 283–295. Springer, 2015.

[82] Fusheng Zha, Yizhou Liu, Xin Wang, Fei Chen, Jingxuan Li, and Wei Guo. Robot motion planning method based on incremental high-dimensional mixture probabilistic model. *Complexity*, 2018, 2018.

[83] Tobias Klamt and Sven Behnke. Towards learning abstract representations for locomotion planning in high-dimensional state spaces. *2019 International Conference on Robotics and Automation (ICRA)*, pages 922–928, 2019.

[84] Anna Yershova, Léonard Jaillet, Thierry Siméon, and Steven M LaValle. Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 3856–3861, 2005.

[85] Kalin Gochev, Benjamin Cohen, Jonathan Butzke, Alla Safonova, and Maxim Likhachev. Path planning with adaptive dimensionality. In *Fourth annual symposium on combinatorial search*, 2011.

[86] Dong-Hyung Kim, Youn-Sung Choi, Taejoon Park, Ji Yeong Lee, and Chang-Soo Han. Efficient path planning for high-DOF articulated robots with adaptive dimensionality. In *IEEE Int. Conf. on Robotics and Automation*, pages 2355–2360, 2015.

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

[88] Ioan A Şucan and Lydia E Kavraki. On the performance of random linear projections for sampling-based motion planning. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference On*, pages 2434–2439. IEEE, 2009.

[89] Constantinos Chamzas, Anshumali Shrivastava, and Lydia E Kavraki. Using local experiences for global motion planning. *2019 International Conference on Robotics and Automation (ICRA)*, pages 8606–8612, 2019.

[90] O Burchan Bayazit, Dawen Xie, and Nancy M Amato. Iterative relaxation of constraints: A framework for improving automated motion planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3433–3440, 2005.

[91] Inho Lee, Jaesung Oh, and HyoIn Bae. Constrained whole body motion planning in task configuration and time. *International Journal of Precision Engineering and Manufacturing*, 19(11):1651–1658, 2018.

[92] Biao Jia, Zherong Pan, and Dinesh Manocha. Fast motion planning for high-dof robot systems using hierarchical system identification. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5140–5147. IEEE, 2019.

[93] Marcus Gary, Nathaniel Fairfield, William C Stone, David Wettergreen, George Kantor, and John M Sharp, Jr. 3d mapping and characterization of sistema zacatón from depthx (de ep p hreatic th ermal e x plorer). In *Sinkholes and the Engineering and Environmental Impacts of Karst*, pages 202–212. American Society of Civil Engineers, 2008.

[94] Hordur Johannsson, Michael Kaess, Brendan Englot, Franz Hover, and John Leonard. Imaging sonar-aided navigation for autonomous underwater harbor surveillance. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4396–4403. IEEE, 2010.

[95] Pedro V Teixeira, Franz S Hover, John J Leonard, and Michael Kaess. Multibeam data processing for underwater mapping. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1877–1884. IEEE, 2018.

[96] Rachel Przeslawski, Scott Foster, Jacquomo Monk, Tim Langlois, VL Lucieer, and RD Stuart-Smith. Comparative assessment of seafloor sampling platforms. report to the national environmental science programme. *report to the National Environmental Science Programme*, 2018.

[97] Arnaud Abadie, Pierre Boissery, and Christophe Viala. Georeferenced underwater photogrammetry to map marine habitats and submerged artificial structures. *The Photogrammetric Record*, 33(164):448–469, 2018.

[98] Pierre Drap. Underwater photogrammetry for archaeology. In *Special Applications of Photogrammetry*. IntechOpen, 2012.

[99] Will Figueira, Renata Ferrari, Elyse Weatherby, Augustine Porter, Steven Hawes, and Maria Byrne. Accuracy and precision of habitat structural complexity metrics

derived from underwater photogrammetry. *Remote Sensing*, 7(12):16883–16900, 2015.

[100] John J Leonard and Hugh F Durrant-Whyte. *Directed sonar sensing for mobile robot navigation*, volume 175. Springer Science & Business Media, 2012.

[101] Fabio Menna, Erica Nocerino, and Fabio Remondino. Photogrammetric modelling of submerged structures: influence of underwater environment and lens ports on three-dimensional (3d) measurements. *In latest developments in reality-based 3D surveying and modelling, MDPI, Basel, Switzerland*, pages 279–303, 2018.

[102] Panagiotis Agrafiotis, Georgios I Drakonakis, Dimitrios Skarlatos, and Andreas Georgopoulos. Underwater image enhancement before three-dimensional (3d) reconstruction and orthoimage production steps: Is it worth. *Latest Developments in Reality-Based 3D Surveying and Modelling; Remondino, F., Georgopoulos, A., González-Aguilera, D., Agrafiotis, P., Eds*, 2018.

[103] Christina Georgiades, Meyer Nahon, and Martin Buehler. Simulation of an underwater hexapod robot. *Ocean Engineering*, 36(1):39–47, 2009.

[104] Philippe Giguere, Chris Prahacs, and Gregory Dudek. Characterization and modeling of rotational responses for an oscillating foil underwater robot. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3000–3005. IEEE, 2006.

[105] Nicolas Plamondon and Meyer Nahon. Trajectory tracking controller for an underwater hexapod vehicle. In *OCEANS 2008*, pages 1–8. IEEE, 2008.

[106] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.

[107] Gregory Dudek, Philippe Giguere, Chris Prahacs, Shane Saunderson, Junaed Sattar, Luz-Abril Torres-Mendez, Michael Jenkin, Andrew German, Andrew Hogue, Arlene Ripsman, et al. Aqua: An amphibious autonomous robot. *Computer*, 40(1):46–53, 2007.

[108] Mark Fiala. Artag, a fiducial marker system using digital techniques. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 590–596, 2005.

[109] Guoquan Huang. Visual-inertial navigation: A concise review. In *Proc. International Conference on Robotics and Automation*, Montreal, Canada, May 2019.

[110] Florian Shkurti, Ioannis Rekleitis, Milena Scaccia, and Gregory Dudek. State estimation of an underwater robot using visual and inertial information. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5054–5060, 2011.

[111] Anastasios I Mourikis and Stergios I Roumeliotis. A multi-state constraint Kalman filter for vision-aided inertial navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3565–3572. IEEE, 2007.

[112] D. Meger, F. Shkurti, D. Cortés Poza, P. Giguère, and G. Dudek. 3d trajectory synthesis and control for a legged swimming robot. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2257–2264, Sep. 2014.

[113] Yinjing Guo, Hui Liu, Xiaojing Fan, and Wenhong Lyu. Research progress of path planning methods for autonomous underwater vehicle. *Mathematical Problems in Engineering*, 2021, 2021.

[114] Clement Petres, Yan Pailhas, Pedro Patron, Yvan Petillot, Jonathan Evans, and David Lane. Path planning for autonomous underwater vehicles. *IEEE Transactions on Robotics*, 23(2):331–341, 2007.

[115] Stefan Williams, Gamini Dissanayake, and Hugh Durrant-Whyte. Towards terrain-aided navigation for underwater robotics. *Advanced Robotics*, 15(5):533–549, 2001.

[116] Bartolomé Garau, Matias Bonet, Alberto Alvarez, Simón Ruiz, and Ananda Pascual. Path planning for autonomous underwater vehicles in realistic oceanic current fields: Application to gliders in the western mediterranean sea. *Journal of Maritime Research*, 6(2):5–22, 2009.

[117] Alberto Alvarez, Andrea Caiti, and Reiner Onken. Evolutionary path planning for autonomous underwater vehicles in a variable ocean. *IEEE Journal of Oceanic Engineering*, 29(2):418–429, 2004.

[118] Namik Kemal Yilmaz, Constantinos Evangelinos, Pierre FJ Lermusiaux, and Nicholas M Patrikalakis. Path planning of autonomous underwater vehicles for adaptive sampling using mixed integer linear programming. *IEEE Journal of Oceanic Engineering*, 33(4):522–537, 2008.

[119] Juan David Hernández, Eduard Vidal, Mark Moll, Narcís Palomeras, Marc Carreras, and Lydia E Kavraki. Online motion planning for unexplored underwater environments using autonomous underwater vehicles. *Journal of Field Robotics*, 36(2):370–396, 2019.

[120] James Ju Heon Lee, Chanyeol Yoo, Raewyn Hall, Stuart Anstee, and Robert Fitch. Energy-optimal kinodynamic planning for underwater gliders in flow fields. In *Australasian Conference on Robotics and Automation, ACRA*, 2017.

[121] Eduard Vidal, Mark Moll, Narcís Palomeras, Juan David Hernández, Marc Carreras, and Lydia E Kavraki. Online multilayered motion planning with dynamic constraints for autonomous underwater vehicles. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8936–8942. IEEE, 2019.

[122] Colin J Green and Alonzo Kelly. Toward optimal sampling in the space of paths. In *ISRR*, pages 281–292. Citeseer, 2007.

[123] Ross A Knepper and Matthew T Mason. Empirical sampling of path sets for local area motion planning. In *Experimental Robotics*, pages 451–462. Springer, 2009.

[124] Ross A Knepper and Matthew T Mason. Real-time informed path sampling for motion planning search. *The International Journal of Robotics Research*, 31(11):1231–1250, 2012.

[125] Mihail Pivtoraiko, Ross A Knepper, and Alonzo Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009.

[126] Michael S Branicky, Ross A Knepper, and James J Kuffner. Path and trajectory diversity: Theory and algorithms. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1359–1364. IEEE, 2008.

[127] Ross A Knepper, Siddhartha S Srinivasa, and Matthew T Mason. Toward a deeper understanding of motion alternatives via an equivalence relation on local paths. *The International Journal of Robotics Research*, 31(2):167–186, 2012.

[128] Juan David Hernández, Mark Moll, Eduard Vidal, Marc Carreras, and Lydia E Kavraki. Planning feasible and safe paths online for autonomous underwater vehicles in unknown environments. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1313–1320. IEEE, 2016.

[129] Eduard Vidal, Juan David Hernández, Narcís Palomeras, and Marc Carreras. Online robotic exploration for autonomous underwater vehicles in unstructured environments. In *2018 OCEANS-MTS/IEEE Kobe Techno-Oceans (OTO)*, pages 1–4. IEEE, 2018.

[130] Lihua Zhu, Xianghong Cheng, and Fuh-Gwo Yuan. A 3d collision avoidance strategy for uav with physical constraints. *Measurement*, 77:40–49, 2016.

[131] Stefan Hrabar. 3d path planning and stereo-based obstacle avoidance for rotorcraft uavs. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 807–814. IEEE, 2008.

[132] Gianluca Antonelli, Stefano Chiaverini, Roberto Finotello, and Riccardo Schiavon. Real-time path planning and obstacle avoidance for rais: an autonomous underwater vehicle. *IEEE Journal of Oceanic Engineering*, 26(2):216–227, 2001.

[133] Mike Eichhorn. A reactive obstacle avoidance system for an autonomous underwater vehicle. In *IFAC world congress*, pages 3–8, 2005.

[134] Lynn R Fodrea and Anthony J Healey. Obstacle avoidance control for the remus autonomous underwater vehicle. *IFAC Proceedings Volumes*, 36(4):103–108, 2003.

[135] David Meger, Juan Camilo Gamboa Higuera, Anqi Xu, Philippe Giguere, and Gregory Dudek. Learning legged swimming gaits from experience. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2332–2338. IEEE, 2015.

[136] T. Manderson and G. Dudek. Gpu-assisted learning on an autonomous marine robot for vision based navigation and image understanding. In *MTS/IEEE OCEANS*, Charleston, SC, USA, Oct. 2018.

[137] Travis Manderson, Juan Camilo Gamboa Higuera, Ran Cheng, and Gregory Dudek. Vision-based autonomous underwater swimming in dense coral for combined collision avoidance and target selection. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[138] Zoltan Csaba Marton, Radu Bogdan Rusu, and Michael Beetz. On Fast Surface Reconstruction Methods for Large and Noisy Datasets. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, May 12-17 2009.

[139] Lening Li. *Birrtopt: A combined software framework for motion planning applied on atlas robot*. PhD thesis, WORCESTER POLYTECHNIC INSTITUTE, 2016.

[140] L. Jaillet, J. Cortes, and T. Simeon. Transition-based rrt for path planning in continuous cost spaces. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2145–2150, Sep. 2008.

[141] S. Skaff, J.J. Clark, and Ioannis Rekleitis. Estimating surface reflectance spectra for underwater color vision. In *British Machine Vision Conference (BMVC)*, pages 1015–1024, Leeds, U.K., Sep. 2008.

[142] Sharmin Rahman, Alberto Quattrini Li, and Ioannis Rekleitis. An Underwater SLAM System using Sonar, Visual, Inertial, and Depth Sensor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1861–1868, Macau, (IROS ICROS Best Application Paper Award Finalist), Nov. 2019.

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

[144] Yogesh Girdhar. *Unsupervised Semantic Perception, Summarization, and Autonomous Exploration for Robots in Unstructured Environments*. PhD thesis, McGill University Libraries, 2015.

[145] Eric Bourque and Gregory Dudek. On the automated construction of image-based maps. *Autonomous Robots*, 8(2):173–190, 2000.

[146] Ioannis Rekleitis, Ai Peng New, Edward Samuel Rankin, and Howie Choset. Efficient boustrophedon multi-robot coverage: an algorithmic approach. *Annals of Mathematics and Artificial Intelligence*, 52(2-4):109–142, 2008.

[147] Howie Choset. Coverage for robotics–a survey of recent results. *Annals of mathematics and artificial intelligence*, 31(1-4):113–126, 2001.

[148] Nare Karapetyan, Kelly Benson, Chris McKinney, Perouz Taslakian, and Ioannis Rekleitis. Efficient multi-robot coverage of a known environment. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1846–1852. IEEE, 2017.

[149] Nare Karapetyan, Adam Braude, Jason Moulton, Joshua A Burstein, Scott White, Jason M O'Kane, and Ioannis Rekleitis. Riverine coverage with an autonomous surface vehicle over known environments. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3098–3104, 2019.

[150] John Aloimonos, Isaac Weiss, and Amit Bandyopadhyay. Active vision. *Int. journal of computer vision*, 1(4):333–356, 1988.

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

[152] Hans Jacob S Feder, John J Leonard, and Christopher M Smith. Adaptive mobile robot navigation and mapping. *The International Journal of Robotics Research*, 18(7):650–668, 1999.

[153] Cyrill Stachniss, Dirk Hahnel, and Wolfram Burgard. Exploration with active loop-closing for fastslam. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 2, pages 1505–1510. IEEE, 2004.

[154] Alexei A Makarenko, Stefan B Williams, Frederic Bourgault, and Hugh F Durrant-Whyte. An experiment in integrated exploration. In *IEEE/RSJ international conference on intelligent robots and systems*, volume 1, pages 534–539. IEEE, 2002.

[155] Ruben Martinez-Cantin, Nando de Freitas, Arnaud Doucet, and José A Castellanos. Active policy learning for robot planning and exploration under uncertainty. In *Robotics: Science and Systems*, volume 3, pages 321–328, 2007.

[156] Ioannis Rekleitis. Single robot exploration: Simultaneous localization and uncertainty reduction on maps (SLURM). In *2012 Ninth Conference on Computer and Robot Vision*, pages 214–220. IEEE, 2012.

[157] Ioannis Rekleitis. Simultaneous localization and uncertainty reduction on maps (SLURM): Ear based exploration. In *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 501–507. IEEE, 2012.

[158] Ioannis Rekleitis. Multi-robot simultaneous localization and uncertainty reduction on maps (MR-SLURM). In *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1216–1221. IEEE, 2013.

[159] Qiwen Zhang, David Whitney, Florian Shkurti, and Ioannis Rekleitis. Ear-based exploration on hybrid metric/topological maps. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3081–3088. IEEE, 2014.

[160] Qiwen Zhang, Ioannis Rekleitis, and Gregory Dudek. Uncertainty reduction via heuristic search planning on hybrid metric/topological map. In *2015 12th Conference on Computer and Robot Vision*, pages 222–229. IEEE, 2015.

[161] Rigoberto Lopez Padilla and Rafael Murrieta-Cid. Maintaining visibility of a landmark using optimal sampling-based path planning. *Computación y Sistemas*, 23(4), 2019.

[162] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Appearance-based active, monocular, dense reconstruction for micro aerial vehicles. *2014 Robotics: Science and Systems Conference*, 2014.

[163] Bryan Penin, Riccardo Spica, Paolo Robuffo Giordano, and François Chaumette. Vision-based minimum-time trajectory generation for a quadrotor uav. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6199–6206. IEEE, 2017.

[164] Riccardo Spica, Paolo Robuffo Giordano, and François Chaumette. Coupling active depth estimation and visual servoing via a large projection operator. *The International Journal of Robotics Research*, 36(11):1177–1194, 2017.

[165] Gabriele Costante, Jeffrey Delmerico, Manuel Werlberger, Paolo Valigi, and Davide Scaramuzza. Exploiting photometric information for planning under uncertainty. In *Robotics Research*, pages 107–124. Springer, 2018.

[166] Matthew Sheckells, Gowtham Garimella, and Marin Kobilarov. Optimal visual servoing for differentially flat underactuated systems. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5541–5548. IEEE, 2016.

[167] Tobias Nägeli, Javier Alonso-Mora, Alexander Domahidi, Daniela Rus, and Otmar Hilliges. Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization. *IEEE Robotics and Automation Letters*, 2(3):1696–1703, 2017.

[168] Tobias Nägeli, Lukas Meier, Alexander Domahidi, Javier Alonso-Mora, and Otmar Hilliges. Real-time planning for automated multi-view drone cinematography. *ACM Transactions on Graphics (TOG)*, 36(4):1–10, 2017.

[169] Ciro Potena, Daniele Nardi, and Alberto Pretto. Effective target aware visual navigation for uavs. In *2017 European Conference on Mobile Robots (ECMR)*, pages 1–7. IEEE, 2017.

[170] Davide Falanga, Philipp Foehn, Peng Lu, and Davide Scaramuzza. Pampc: Perception-aware model predictive control for quadrotors. In *2018 IEEE/RSJ In-*

ternational Conference on Intelligent Robots and Systems (IROS), pages 1–8. IEEE, 2018.

[171] Lingjie Yang, Zhihong Liu, Xiangke Wang, and Yinbo Xu. An optimized image-based visual servo control for fixed-wing unmanned aerial vehicle target tracking with fixed camera. *IEEE Access*, 7:68455–68468, 2019.

[172] Ciro Potena, Daniele Nardi, and Alberto Pretto. Joint vision-based navigation, control and obstacle avoidance for uavs in dynamic environments. In *2019 European Conference on Mobile Robots (ECMR)*, pages 1–7. IEEE, 2019.

[173] Keuntaek Lee, Jason Gibson, and Evangelos A Theodorou. Aggressive perception-aware navigation using deep optical flow dynamics and pixelmpc. *IEEE Robotics and Automation Letters*, 5(2):1207–1214, 2020.

[174] Varun Murali, Igor Spasojevic, Winter Guerra, and Sertac Karaman. Perception-aware trajectory generation for aggressive quadrotor flight using differential flatness. In *2019 American Control Conference (ACC)*, pages 3936–3943. IEEE, 2019.

[175] Igor Spasojevic, Varun Murali, and Sertac Karaman. Perception-aware time optimal path parameterization for quadrotors. *International Conference on Robotics and Automation (ICRA)*, pages 3213–3219, 2020.

[176] Melissa Greeff, Timothy D Barfoot, and Angela P Schoellig. A perception-aware flatness-based model predictive controller for fast vision-based multirotor flight. *IFAC-PapersOnLine*, 53(2):9412–9419, 2020.

[177] Boyu Zhou, Jie Pan, Fei Gao, and Shaojie Shen. Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight. *IEEE Transactions on Robotics*, 37(6):1992–2009, 2021.

[178] Zichao Zhang and Davide Scaramuzza. Fisher information field: an efficient and differentiable map for perception-aware planning. *arXiv preprint arXiv:2008.03324*, 2020.

[179] Sergey Frolov, Bartolame Garau, and James Bellingham. Can we do better than the grid survey: Optimal synoptic surveys in presence of variable uncertainty and decorrelation scales. *Journal of Geophysical Research: Oceans*, 119(8):5071–5090, 2014.

[180] Travis Manderson, Juan Camilo Gamboa Higuera, Stefan Wapnick, Jean-François Tremblay, Florian Shkurti, David Meger, and Gregory Dudek. Vision-based

goal-conditioned policies for underwater navigation in the presence of obstacles. *Robotics: Science and Systems XVI*, Jul 2020.

[181] Sharmin Rahman, Alberto Quattrini Li, and Ioannis Rekleitis. Contour based reconstruction of underwater structures using sonar, visual, inertial, and depth sensor. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8048–8053, Macau, Nov. 2019.

[182] Md Modasshir, Alberto Quattrini Li, and Ioannis Rekleitis. Mdnet: Multi-patch dense network for coral classification. In *OCEANS 2018 MTS/IEEE Charleston*, pages 1–6. IEEE, 2018.

[183] Md Modasshir, Sharmin Rahman, Oscar Youngquist, and Ioannis Rekleitis. Coral identification and counting with an autonomous underwater vehicle. In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 524–529. IEEE, 2018.

[184] Md Modasshir, Sharmin Rahman, and Ioannis Rekleitis. Autonomous 3d semantic mapping of coral reefs. In *12th Conference on Field and Service Robotics (FSR), Tokyo, Japan*, 2019.

[185] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[186] Andreas Orthey, Sohaib Akbar, and Marc Toussaint. Multilevel motion planning: A fiber bundle formulation. *arXiv preprint arXiv:2007.09435*, 2020.

[187] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer science & business media, 2013.

[188] Marios Xanthidis, James Johnson, Jason M O'Kane, and Ioannis Rekleitis. Towards multi-camera active perception of underwater structures. In *Proceedings of Advanced Marine Robotics Technical Committee Workshop on Active Perception at IEEE International Conference on Robotics and Automation*, 2021.