FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Planning search missions using a small size AUV

Nuno Miguel Neves de Abreu

Doctoral Program in Electrical and Computer Engineering

Supervisor: Aníbal Castilho de Coimbra Matos (PhD)

September, 2017

Abstract

Autonomous underwater vehicles (AUVs) are increasingly being used to perform search operations but its capabilities are limited by the efficiency of the planning process. Underwater or surface vehicles are able to perform search and rescue missions with higher efficiency, reduced search time and even with covert search capability when compared to traditional methods. The objective of the thesis is to propose new survey planning methods for AUVs. In particular, the problem of multi-objective search mission planning with an AUV navigating in known or unknown 3D environments is studied. The vehicle should completely cover the operating area while maximizing the probability of detecting the targets and minimizing the required energy and time to complete the mission.

Typical coverage path-planning (CPP) approaches assume either a perfect knowledge of the environment, a simplistic image of the environment or no uncertainty in sensing or control. These are unrealistic assumptions in the vast majority of real-world scenarios and especially in the challenging underwater environment.

A multi-stage algorithm is proposed and evaluated. Our algorithm combines an evolutionary algorithm (EA) with a local search procedure, aiming at a more flexible and effective exploration and exploitation of the search space. This approach is an alternative to classical optimization methods with the capability of solving problems involving a big search space quickly, although not guaranteeing that an optimal solution is found. One of its most important features is their capability of finding a distinct set of best solutions, a Pareto Front, optimizing each of the objectives in a different manner.

An artificial neural network (ANN) model was also integrated in the evolutionary procedure to guide the search. The increasing dominance that the Pareto set generated by the EA using the ANN exhibited is a consequence of an improving representation of the search space, maintained by the ANN training dataset, as the search progresses. Therefore, this proves that the integration of the ANN is advantageous to the search process since it is guiding the search to more promising regions.

The basic idea behind the local optimization algorithm is to try to locally improve the Pareto Front found by the EA. This is achieved by choosing the best set of inter-track distances maximizing the coverage of an area characterized by a specific topography. Distinct strategies are used depending on the type of sonar that is being used for seafloor mapping. Results show that the average probability of detection (POD) increased and that the amount of uncovered cells was reduced. In addition, it was proven that the algorithm could successfully cover the nadir gaps, featured by the sidescan sonar.

The combination of different techniques creates another problem, related to the high amount of parameters that need to be tuned. Thus, the effect of these parameters on the quality of the obtained Pareto Front was assessed. This allowed us to define an adaptive tuning procedure to control the parameters while the algorithm is executed.

Our planning algorithm was compared against an implementation of a known EA as well as another mission planners and the results from the experiments show that the proposed strategy can efficiently identify a higher quality solution set and produce better results in real-world scenarios.

As mentioned previously, offline CPP algorithms described in literature assume that the map used to plan the path accurately represents the environment and that the vehicle will be able to precisely execute the path, ignoring any uncertainty in sensing or control. However, these assumptions may not hold when operating in a challenging environment, where the robot's position estimation is subject to error, perturbations such as currents affect its control actions and the acoustic sensors used to map the environment might detect inaccuracies in the prior map. The approach presented here differs from other CPP methods in that paths for coverage are generated based on a coverage map that is actively maintained as the vehicle executed its mission. Our replanning approach borrows ideas from case-based reasoning in which old problem and solution information, more precisely the solutions from the Pareto Front given by the offline planner, helps solve a new problem. The resulting combination takes advantage of both paradigms where our evolutionary approach in conjunction with ANNs, presented earlier, delivers robustness and adaptive learning while the case-based component speeds up the replanning process. The online replanner generates a new mission plan by one of the following alternatives: locally optimizing the current one; evaluating the cases in the casebase according to the new aquired information and choose the best; when no acceptable mission plan can be retrieved, executing the evolutionary planner to perform exhaustive search until one is found. The task of achieving complete coverage in unknown environments consists of many tasks such as: trajectory estimation (to understand where the vehicle has been), map building (to update the representation of the environment), clustering (to identify uncovered regions and thus future replanning areas) and the replanning process itself, accounting for all the new data. The algorithms are tested in simulation and in the field on the MARES AUV. The experiments show that the online algorithm was able to successfully replan missions in varied scenarios and guarantee full area coverage while minimizing resource consumption.

Additionally, a coverage path planning technique for search operations is presented which takes into account the vehicle's position and detection performance uncertainties and tries to minimize this uncertainty along the planned path. The objective is to plan paths, using a localization error model as input, to reduce as much uncertainty as possible and to minimize the extra path length (swath overlap) while satisfying mission feasibility constraints. An algorithm that calculates what will be the best moments for bringing the vehicle to surface to ensure a bounded position error is also introduced.

Resumo

Os veículos autónomos submarinos estão cada vez mais a ser utilizados para realizar operações de busca mas as suas capacidades estão limitadas pela eficiência do processo de planeamento. Estes veículos são capazes de realizar operações de busca com maior eficiência, num menor período de tempo e até com algum grau de secretismo quando comparado com os métodos tradicionais. O objetivo desta tese é propor novas técnicas de planeamento de operações de busca com AUVs. Em particular, o problema abordado aqui é o do planeamento de missões de busca multi-objetivo com AUVs, navegando em ambientes tridimensionais conhecidos ou desconhecidos. O veículo deverá cobrir completamente a área de operação, procurando maximizar a probabilidade de detetar os objetos e minimizando a energia e o tempo necessários para completar a missão.

Os métodos de planeamento de trajetórias de cobertura tradicionais assumem ou o conhecimento perfeito do ambiente, uma imagem simplificada deste ou a inexistência de incerteza na amostragem e no controlo. Estas são assunções irrealistas na maioria dos cenários encontrados no mundo real e especialmente no ambiente subaquático.

Como resposta a estas questões foi proposto e avaliado um algoritmo multi-etapas. O algoritmo combina um algoritmo evolucionário com um método de otimização local, procurando aumentar a flexibilidade e eficiência da exploração do espaço de pesquisa. Esta abordagem é uma alternativa aos métodos clássicos com a capacidade de resolver mais rapidamente problemas envolvendo grandes espaços de pesquisa, mas não garantindo que uma solução ótima é encontrada. Uma das características mais importantes é a capacidade de encontrar um conjunto distinto de melhores soluções, a frente de Pareto, otimizando cada um dos objetivos considerados de maneira diferente.

Um modelo de rede neuronal artificial foi também integrado no algoritmo evolucionário para guiar a pesquisa. A dominância crescente do conjunto de Pareto gerado pelo algoritmo que usa a rede neuronal é consequência de uma melhor representação do espaço de pesquisa, mantido pelo conjunto de treino da rede, à medida que a pesquisa evolui. Então, isto demonstra que a integração da rede neuronal é vantajosa para o processo de pesquisa uma vez que a está a guiar para melhores regiões.

A ideia por detrás do algoritmo de otimização local é tentar melhorar localmente a frente de Pareto encontrada pelo algoritmo evolucionário. Isto é atingido escolhendo o melhor conjunto de espaçamentos entre os segmentos paralelos de forma a maximizar a cobertura da região considerada, tendo em conta a sua topografia. São adotadas estratégias distintas dependendo do tipo de sonar que vai ser usado para fazer o varrimento do fundo. Os resultados mostram um aumento da probabilidade de deteção média e uma diminuição da quantidade de área não coberta. Além disto, foi provado também que o algoritmo é capaz de cobrir as regiões de *nadir*, características do sonar de varrimento lateral.

A combinação de diferentes técnicas cria outro problema relacionado com a grande quantidade de parâmetros que têm de ser ajustados. Por esta razão, também foi determinado o efeito destes parâmetros na qualidade da frente de Pareto obtida. Isto permitiu definir um procedimento de

ajuste adaptativo para controlar os valores destes parâmetros durante a execução do algoritmo evolucionário.

Este método de planeamento foi comparado com implementações de outros algoritmos evolucionários assim como com outros planeadores de missões que envolvem cobertura. Os resultados das experiências mostram que o método proposto consegue identificar um conjunto de soluções superior, e como consequência, produzir melhores resultados em cenários realistas.

Tal como foi referido anteriormente, os métodos de planeamento *offline* descritos na literatura assumem a existência de uma mapa que representa o ambiente de forma precisa e que o veiculo é capaz de seguir exatamente a trajetória planeada, sem incerteza na amostragem ou no controlo. No entanto, estas assunções podem deixar de ser válidas quando se opera em ambientes realistas onde o sistema de posicionamento do veículo está sujeito a erros, perturbações como correntes que afetam o controlo e a performance dos sensores usados para fazer o mapeamento. A abordagem apresentada aqui difere de outros métodos *online* na forma como são geradas as trajetórias, baseadas num mapa que é atualizado durante a missão. Este método de replaneamento usa ideias da técnica de raciocínio baseado em casos, onde problemas passados e as suas soluções (neste caso são a frente de Pareto) ajudam a resolver o novo problema de forma mais eficiente. Esta combinação com os algoritmos evolucionários e a rede neuronal junta vantagens de ambos os paradigmas, constituindo um método de aprendizagem robusto e adaptativo com a capacidade de acelerar o processo de replaneamento. O replaneador *online* gera uma nova missão de acordo com uma das seguintes estratégias: otimização do plano atual; avaliação e adaptação de casos anteriores e escolha do melhor; em ultimo caso, quando não for possível gerar um plano aceitável, voltar a executar o algoritmo evolucionário para realizar uma pesquisa exaustiva até ser encontrada uma solução. O processo de replaneamento consiste em diversas fases tais como: estimação da trajetória, construção do mapa topográfico, identificações de regiões não cobertas e replaneamento da missão tendo em conta toda a nova informação. Os algoritmos foram testados tanto em simulações como no campo com o veiculo MARES. Estes testes mostraram que a abordagem *online* conseguiu replanear missões com sucesso em cenários variados e garantir cobertura total da área de operação, minimizando a utilização dos recursos.

Adicionalmente, foi apresentada uma técnica de planeamento de trajetórias para cobertura que tem em conta a performance do sistema de posicionamento e deteção do veiculo, tentando minimizar a incerteza ao longo do caminho. O objetivo é planear trajetórias usando um modelo do erro de localização como variável de entrada para reduzir a incerteza tanto quanto possível, minimizando o comprimento adicional do caminho e respeitando as restrições da missão. O algoritmo desenvolvido calcula também quais os melhores momentos para trazer o veiculo à superfície, permitindo reduzir a incerteza no sistema de localização ao captar mensagens de GPS.

Acknowledgements

It has been a long time since I first arrived at the Electrical Engineering department in the Faculty of Engineering at the University of Porto as an undergraduate. I am proud of having had the opportunity to study at this institution because it has made me the engineer I am today and gave me all the tools and confidence I need to solve any real-world problem. This has been a wonderful place to work at. Now I want to thank some of the people who helped make it an good experience.

First of all, I would like to express my genuine gratitude to my supervisor Dr. Aníbal Matos. His encouragement for me to pursue my PhD was decisive and his contribution to my work, as researcher and engineer, was invaluable. Without his support, guidance, and specially funding, this thesis would simply not have been possible. I also appreciate the patience he had and the freedom he gave me to dedicate myself to this project, since I had to delay the execution of many tasks I was responsible for in our research group as a consequence.

I would like to thank the Institute for Systems and Computer Engineering, Technology and Science (INESC TEC) who have funded my studies, allowed me to work in several interesting projects and travel to conferences all around the world.

To all of my friends in the underwater robotics research group (Oceansys) over the years: Rui, Bruno, André, José and professor Nuno, it has been a pleasure working and facing all the (difficult) challenges together with you. A special mention should go to Bruno Ferreira, who helped me getting started with the simulator and solve all those issues.

Finally, my family. My parents Margarida and Wilson, provided me all the love, education and support one could hope for. I grew up seeing them working hard for getting their MSc and PhD degrees and never really thought I was capable of such achievement. But they taught me the value of hard work and always challenged me to do my best, so I dedicate this work to them.

Nuno Abreu

Contents

List of Figures

List of Tables

Abbreviations

ANN	Artificial neural network
AUV	Autonomous underwater vehicle
BCD	Boustrophedon cellular decomposition
CAS	Collision avoidance system
CBR	Case-based reasoning
CPA	Closest point of approach
CPP	Coverage path planning
DEM	Digital elevation model
DVL	Doppler velocity log
EA	Evolutionary algorithm
EKF	Extended Kalman Filter
EKS	Extended Kalman Smoother
FLS	Forward looking sonar
GIS	Geographic information system
GPS	Global positioning system
IMU	Inertial measurement unit
INS	Inertial navigation system
LRC	Lateral range curve
MCM	Mine countermeasures
MOP	Multi-objective problem
MOEA	Multi-objective evolutionary algorithm
NSGA	Nondominated sorting genetic algorithm
POD	Probability of detection
POS	Probability of success
RL	Reinforcement learning
ROV	Remotely operated vehicle
SAR	Search and rescue
SLAM	Simultaneous localization and mapping
SPEA	Strength Pareto evolutionary algorithm
SSS	Sidescan sonar
UDP	User datagram protocol
UAV	Unmanned aerial vehicle
USV	Unmanned surface vehicle

Chapter 1

Introduction

1.1 Addressed problem

The push for automation in everyday tasks has caused mobile robots to integrate our life, increasing its usage in the civilian and military domains. Either in the form of aerial, maritime or ground vehicles, robots are already employed within challenging operations such as search and rescue (SAR), environmental monitoring, structure inspection, exploration and surveillance. Among all possible different robot configurations, AUVs have attracted significant attention in the last several years. One of the driving forces has been the offshore industry and the need for fast and accurate surveying and inspection. AUVs have also become increasingly important for oceanography. The oceans are virtually unexplored and the use of AUVs is proving to be the best option for surveying large areas. They are able to provide high resolution maps of the sea floor, requiring little human intervention compared to their ship or remotely operated vehicle (ROV) assisted counterparts, and hence operate at a lower cost during many hours. Underwater or surface vehicles are able to perform search and rescue missions with higher efficiency, reduced search time and even with covert search capability when compared to traditional methods. Larger mission areas can easily be surveyed by deploying several vehicles for significant periods of time.

At present, in most AUV survey missions, the vehicle passes a down-looking sensor over the seafloor by following a preplanned path with a lawnmower pattern while keeping a safe altitude from the bottom. However, navigating at a conservative altitude imposes important limitations for a number of emerging applications demanding high resolution seafloor surveys of rugged terrain or close object inspections. Examples include monitoring of cold water coral reefs (Grasmueck et al., 2006), oil and gas pipeline inspection (Petillot et al., 2002), harbor (Kermorgant, 2005) and dam protection (Cruz et al., 2011), and object recovery (Fernández et al., 2013).

Nonetheless, fully autonomous AUV surveys still have important limitations. Such operations require the AUVs to be submerged for extended periods of time and that they possess high degree of autonomy. The basic feature of an autonomous robot is its capability to operate on its own in unknown or partially known environments. This autonomy implies that the robot is capable of reacting to static obstacles and unpredictable dynamic events that may block the successful

execution of a task. To achieve this goal, solutions need to be developed in path planning, map building and navigation. Such path planning algorithms should provide quick computation of efficient paths that result in full coverage of the area or structure to be inspected, while respecting the onboard sensor limitations as well as the vehicle motion constraints that may apply. The path planning process that involves passing a robot's sensor over all points in a target area is known as coverage path planning (CPP), and there is substantial research addressing this problem in the literature (see next chapter). The task of covering a bounded region of space is an essential component of many robotic activities such as bathymetry, humanitarian demining, and search and rescue. For example, a demining robot must scan for the presence of mines over the entire terrain that is not obstructed by obstacles (Acar et al., 2003). In such applications, it is essential to ensure the completeness of coverage, meaning that no accessible area should be left uncovered at the end of the coverage mission.

Typical CPP approaches assume perfect knowledge of the environment and no uncertainty in sensing or control. These are unrealistic assumptions in the vast majority of scenarios and especially in underwater environments, even when using techniques such as terrain-relative navigation (TRN) or simultaneous localization and mapping (SLAM) for reduced localization uncertainty. This limits real world application of those techniques to constrained environments.

Developing reliable path planning algorithms that help AUVs achieve complex mission objectives without any human intervention is a serious challenge. The restrictions on the path planning module are usually severe since it is operating within a larger system with limited onboard resources. For example, the path planning algorithm must be able to plan one or more safe paths with minimum energy expenditure that fulfils the objective in a timely manner based on acquired sensor data without hogging system resources such as CPU cycles or memory.

Solving a multi-objective path planning problem is computationally very expensive. The upper bound on the complexity of a n degree-of-freedom path planning problem is $O(n^n)$, meaning that the complexity of the path planning problem grows exponentially with the number of degrees-of-freedom (Canny, 1988). EAs have been successfully used in the past for solving the multi-objective path planning problem (Fujimura, 1996, Xiao et al., 1997). This approach is an alternative to classical optimization methods with the capability of solving problems involving a big search space quickly, although not guaranteeing that an optimal solution is found. One of its most important features is their capability of finding a distinct set of best solutions, optimizing each of the objectives in a different manner. The analysis of each of these solutions in terms of trade-offs provides a better understanding of the problem to the decision maker.

It is assumed that an environment where an AUV navigates in 3D space is partially known. Although mapping data of the seafloor is given in advance, it is likely incomplete or inaccurate as there may exist unknown obstacles in the area, for example. Therefore, knowledge about the environment must be incrementally acquired by using sensors onboard the AUV. Path planning in this context is defined as follows:

- Offline path planning: Given elevation data and an operation area, compute a collision-free path that completely covers the area so that the path is optimized with respect to the available

information and objectives;

- Online path planning: Whenever new information is acquired due to incomplete information about the environment or when erratic behaviour of the vehicle is detected, update the current path accordingly.

For this reason, the performance of the AUV in navigating in an unknown environment depends strongly on its perception capabilities. It needs to acquire a large amount of data using its sensors and to integrate it into a proper representation of the environment. The traditional paradigm for representing spatial information is based on the use of a 2D tessellation called an occupancy grid (Elfes, 1990), which stores qualitative information about which areas of the robot's surroundings are empty and which areas are occupied by obstacles.

Map building using sonars has been addressed by many researchers and a substantial body of modelling and experimental work has been presented (Johnson-Roberson et al., 2010, Roman and Singh, 2007, Yoerger et al., 1999). The sensor provides the relative distance between the robot and surrounding obstacles located within the sensor footprint. The time elapsed between the transmission of a acoustic signal and the reception of its echo allows the computation of a range reading. This means that an obstacle may be located somewhere on the circumference with radius equal to the measured range. However, these devices are prone to several measuring errors due to various phenomena, for example, multiple reflections and low angular resolution. As a consequence, the sensing process is affected by a large amount of uncertainty. In order to handle uncertainty during the grid building process, a data fusion approach is used to process the data and build an updated map.

Many real world problems in robotics are dynamic and require dynamic algorithms capable of adapting over time. Algorithms may need to react not only to changes in the physical environment, but also to dynamic vehicle and mission constraints. An optimal solution for one instance might not be optimal in the next instance. Therefore, there is a need for robots with the ability to learn, memorize, and deal with different environments. The development of learning and memory facilities for autonomous robots constitutes one of the major trends in the current research on robotics (Connell and Mahadevan, 1993) . Researchers have found long ago that, to build a sufficiently autonomous robot capable of interacting with real world applications, both learning and memorizing algorithms should be combined and concurrently introduced to the robot (Moore, 1990). This is the only mechanism that allows the robot to efficiently use its experience to increase the effectiveness of its adaptive ability. Moreover, it may help reducing the cost of programming for specific tasks. Such a feature creates a memory problem related to how to efficiently handle past experiences. Some implemented a kind of memory with a forgetting mechanism (Kira and Arkin, 2004, Kruusmaa, 2003). This mechanism, however, can affect negatively the robot's performance in some domains.

The major challenge that will drive our research is the design and development of a reliable and autonomous system necessary to execute 3D surveys with different sensor payloads having search applications in mind. This requires advances at different mission phases of an AUV operation:

- Pre-mission: advanced coverage planning algorithms should take into account already available elevation maps, the characteristics of the imaging sensors and AUV kinematic constraints, as well as procedures for the optimization of the navigation, mapping and communications, to generate optimal survey trajectories;
- During-mission: navigation in an unconfined environment still is a major challenge; online coverage planning algorithms will be needed to improve the offline planned paths given new information acquired during mission execution; navigation and safe guidance are also challenging, since for coverage applications navigating with minimum uncertainty is a requirement; mapping using different sensor types (forward looking sonar, sidescan sonar, altimeter) and their fusion into a single consistent map is also a challenge, specially considering the uncertainty in navigation data; handling all these capabilities using the onboard computer is in itself also a challenge given the limited processing power and the complexity of the problem;
- Post-mission: how to combine data coming from different sensors and even vehicles into a single and consistent model of the environment is one of the main challenges; finally, new methods will be needed to extract and classify relevant information from within the large amount of data acquired by the vehicle, for human interpretation and improvement of the processes.

Due to the nature of the coverage problem, and despite the efforts of the community, its inherent difficulties still pose limitations on the performance, efficiency and practical applicability of the proposed solutions especially when navigation performance on complex environments is considered. Furthermore, to the author's best knowledge, only few works validated their algorithmical contributions with real life experiments, assessed their ability to be implemented onboard and estimated coverage quality using the robot's sensors.

1.2 Thesis outline

This thesis consists of eight chapters. Chapter 2 describes the coverage path planning problem in detail. The concept of "path planning" is explored by presenting an overview of the traditionally used techniques for path planning and discussing some of its advantages and disadvantages. In particular, previously proposed techniques for coverage path planning, used in different applications, are described. Other subjects which are essential to the completeness of this case study are also discussed, such as the measures of performance that are typically considered in search operations and the main features of the sensors used for performing the these missions. Chapter 3 specifies the problem under study. Chapter 4 presents our vehicle, the MARES AUV, the simulator and details new developments added while performing this study. More specifically, the sonar simulator, the inertial navigation system, the collision avoidance system and the MARES AUV energy model are described. Chapter 5 describes our proposed multi-objective procedures for offline mission planning. The underlying mechanisms that guide the EA are described, as well as

the local optimization technique. This algorithm is also compared with other existent approaches. Chapter 6 extends the approach of chapter 5 by adding the capability of online mission replanning. Several strategies for mission replanning are discussed and experiments are performed to demonstrate its functionality. The behaviour of our algorithms is analysed and some of its problems are identified. A methodology for accounting for navigation errors is described in chapter 7. This can be used as an add-on to other CPP techniques. Chapter 8 presents the conclusions and suggests some of the areas that are going to be the subject of future research, hopefully contributing to the elimination of the weaknesses of the proposed algorithms and to the addition of useful and innovative capabilities.

1.3 Contributions

The key contributions of this work include:

- An multi-objective multi-stage approach combining a EA with simulated annealing for planning search operations in static 3D environment with known terrain. Our algorithm maintains a diverse population of feasible solutions, integrates an ANN in order to explore the search space and uses simulated annealing to improve the best solutions found. It is the first mission planning technique that maximizes detection performance while minimizing energy consumption and the time required to perform the mission in a complex environment.

- A mission replanning technique that uses past experience to decrease replanning time while maximizing detection performance and minimizing resource consumption. Past experience is maintained in the form of previous solutions generated by the EA and its ANN. The algorithm is capable of handling unknown environments by first executing a conservative exploratory mission and then process all the acquired information and replan accordingly. It includes map building and clustering techniques and the replanning strategy is chosen by evaluating performance of each of these phases.

- A CPP technique for search operations which takes into account the vehicle's position and detection performance uncertainties and tries to minimize this uncertainty along the planned path. It accounts for deviations from the preplanned path due to navigation errors. The algorithm also calculates what will be the best moments for bringing the vehicle to surface to ensure a bounded position error by getting a position fix.

- Field trials and simulations which demonstrate that the proposed methods are successful and effective in planning search missions.

Other important contributions, that do not constitute original research but where a great amount of time was spent working on and will be the base for future developments, are:

- the sonar simulator, using ray-tracing techniques;

- the collision avoidance system;

- the inertial navigation system.

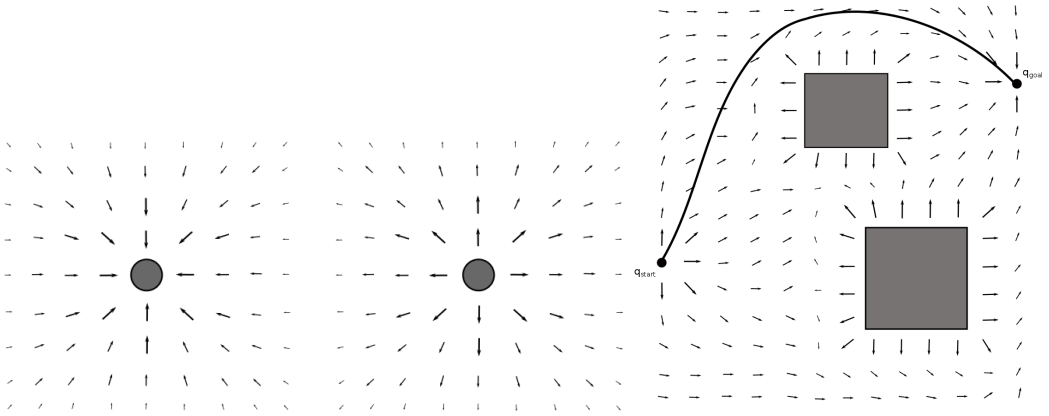
Chapter 2

Related Work

2.1 Path planning in robotics

The basic motion planning problem deals with static environments, that is, workspaces solely containing stationary obstacles of which the geometry is known. Approaches to this problem are often divided into three main approaches, originally proposed by Latombe (Latombe, 1990):

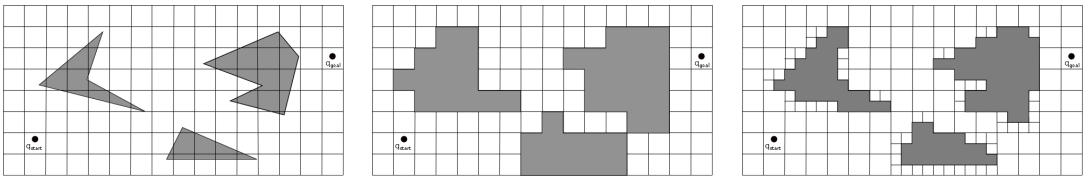
- The potential field approach uses an artificial potential function to guide the robot. The goal configuration pulls the robot by generating an attractive potential and the obstacles repel the robot by generating a repulsive potential to avoid collisions. The resultant potential field is obtained by combining these potentials. As there is little preprocessing, this approach has been popular in scenarios requiring online real-time navigation. But the robot often ends up in a local minimum of the potential.



(a) Attractive potential (b) Repulsive Potential (c) The Potential Field concept: The obstacles and the start and goal configurations exert forces into the robot; the magnitude is proportional to the distance from this locations.

Figure 2.1: Potential fields.

- A cell decomposition approach first divides the free configuration space into simple, non-overlapping geometric cells. The adjacency relations between the cells are then constructed using a connectivity graph. Motion planning is then reduced to the problem of finding a sequence of cells (known as channel) that connects the cell containing the initial configuration to the cell containing the goal configuration.



(a) The configuration space converted into a regular grid.

(b) Decomposition of the configuration space in equal squares with small dimensions and corresponding classification as $free$ or $obstructed$.

(c) Adaptive cell decomposition. *Mixed* cells, partially occupied by obstacles, are subdivided until a minimum cell size is reached. This enables the extraction of extra free configuration space.

Figure 2.2: Approximate cell decomposition (Brooks and Lozano-Perez, 1985).

- The roadmap approach consists in mapping the robot's environment and representing the connectivity of the free configuration space. A roadmap is a class of topological map that represents the environment using graph like structures, where the nodes in the graph are configurations in the free configuration space and the edges indicate a simple path connecting configurations. There are several types of roadmaps and methods for computing them: visibility graphs, the Voronoi diagram and the silhouette method.

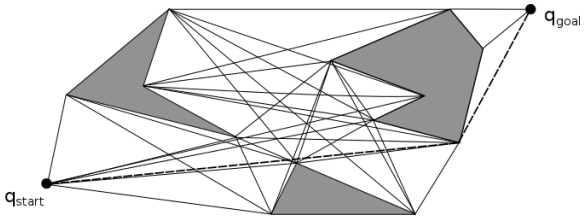


Figure 2.3: Visibility Graph. The shortest path must go through some of the vertices of the obstacles.

Extracting reliable information about the environment dynamics is extremely important when a mobile robot is intended to navigate through the unknown. It is then understandable why sampling-based path planners are the most successful algorithms for planning in such spaces. Sampling-based path planners, like the probabilistic roadmaps (PRM)(Kavraki, 1994) and rapidly-growing random trees (RRT)(Lavalle, 1998), build an approximate representation of configuration space connectivity. Instead of trying to discretize the configuration space and then performing intensive exploration, sampling-based approaches rely on particular sampling strategies to sample and examine configurations. Since this is a stochastic process, sampling-based algorithms are considered

probabilistically complete. This means that the probability of finding a solution or determining that no solution exists goes to one as time goes to infinity (without timing constraints all configuration space can be covered). Therefore, the probability of finding a solution can be made arbitrarily large by acquiring a sufficient number of samples.

After representing the free configuration space using a graph or tree, graph search algorithms needs to be executed to analyse every node and determine the best path. Several search methods have been developed for graph searching, such as the classical Depth-first, Breadth-first, Best-first, A*, and Dijkstra's shortest path algorithms, or the advanced biologically inspired search techniques such as particle swarm optimization and genetic algorithms. These methods were detailed on the thesis research plan presented earlier but, since the main focus of this thesis is on coverage path planning, it was decided to just present a brief summary of the original chapter.

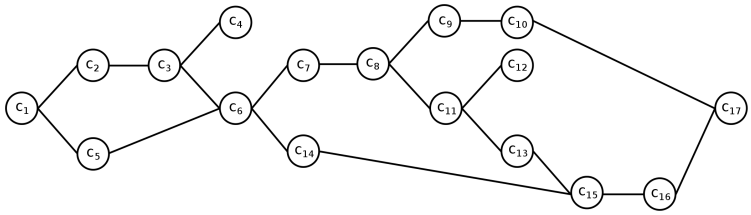


Figure 2.4: Connectivity graph representing the adjacency relation among cells. Once the cells that contain the start and goal are determined, we need a algorithm to search the associated connectivity graph to determine the best path between the cells.

2.2 Coverage path planning

Coverage strategies can be classified according to several criteria. One distinctive characteristic is whether a certain technique needs access to a map of the environment prior to performing coverage, or if it is capable of achieving coverage of an unknown environment. This characterization is often differentiated as offline coverage versus online coverage (Choset, 2001).

An offline planner assumes that perfect prior knowledge of the workspace exists. An efficient coverage algorithm for known environments should be able to create a coverage path with the shortest amount of traveling time or with the least energy expenditure, depending on the specific requirements of the application domain. The distinction between time and energy has practical importance when operating under the influence of external disturbances, for example, when operating an AUV in the presence of strong ocean currents. This distinction will be highlighted in the in the course of the current work.

While the survey is being performed by the autonomous vehicle there may be a need to decide whether to continue with preplanned (offline) search paths or recalculate a new path in the presence of new information. If the autonomous vehicle is supposed to have the ability to adapt to the environment then an online planner is required. In unknown environments, obstacles have to be detected while operating and should be considered by the path planner. This requires environmental sensing, mapping, and fast path replanning during mission execution.

CPP is related to the classical start-to-goal path planning objective with notable differences. Consider a mobile robot with an onboard sensor with some 2D swath S. A trajectory through the workspace W will result in a set of N sensor readings: $S_1, S_2, ..., S_K$. The goal of CPP is to generate a path through the workspace such that:

$$\cup_{k=1}^{K} S_k \supseteq W \tag{2.1}$$

meaning that at some point the sensor swath has passed over all points in the workspace. Each swath is ideally represented by two waypoints, forming a straight line transect.

2.2.1 Planning in 2D environments

A large body of research has investigated coverage path planning in simple 2D environments. Butler et al. (Butler et al., 1999) developed an algorithm based on the incremental construction of a cell decomposition of an unknown environment. This allows a robot with perfect position sensing to cover environments with rectilinear boundary and obstacles. The robot's exploration is directed by the current state of the cell decomposition and its position, continuing until there is no part left unknown. The choice of direction is determined by an ordered list of rules, where the preferred behaviour is the one described by the lawn-mowing pattern. Other rules are described to deal with obstacles and boundaries. They proved the completeness of the algorithm analytically and experimentally through simulation.

Choset (Choset, 2000) developed a coverage algorithm based on a decomposition technique named boustrophedon decomposition, assuming complete knowledge about the environment. In this method, a line segment is swept through the environment and whenever there is a change in connectivity of the slice, a new cell is formed. Once the decomposition and adjacency graph are determined, the robot employs a simple graph search algorithm to determine the coverage order. Complete coverage is achieved by visiting each cell using the lawn-mowing pattern.

Acar and Choset (Acar and Choset, 2002a, Acar et al., 2002) updated the algorithm previously developed by Choset (Choset, 2000) to cover unknown environments using Morse Decompositions. The robot performs coverage and detects critical points during the process. Experiments showed that algorithm fails in some cases as localization uncertainty is not considered. It also cannot handle surfaces parallel to the sweep line. This excludes rectilinear environments.

A few years later, Acar and Choset (Acar et al., 2006) updated their algorithm for unknown area coverage by considering a detector with finite range. Coverage is achieved in two steps: first perform area coverage in open spaces using full detector range employing previous work in using Morse cell decompositions; second step considers area coverage in tight or cluttered spaces with obstacles. Here, the detector is used to accumulate information about the environment and then the robot can cover the area by following the generalized Voronoi diagram (GVD) of that space, built online.

Wong and MacDonald (Wong and MacDonald, 2004) proposed a exact cell decomposition algorithm, known as slice decomposition, based on topological structures of the environment. A

slice decomposition is created by sweeping a line from the top of the environment to the bottom, where the sweep lines represent the long strips in the zigzag motion. At anytime, the sweep line intersects a number of free space and obstacle regions determined by the topology of the environment and position of the sweep line. Since slice decomposition uses a discrete line sweep process, the step size between consecutive slices therefore affects the decomposition performance for a given environment. In practice the step size is determined by the width of the robot, to ensure no space is left uncovered in consecutive sweeps. The decomposition results are presented as a topological map, and each cell is covered by the robot via simple zigzag motion. It can be constructed online, in an unknown environment, while the robot covers the space (Wong and MacDonald, 2003).

Maza and Ollero (Maza and Ollero, 2007) proposed a 2D terrain coverage technique using a team of heterogeneous UAVs. The algorithm only assumes convex areas and no obstacles. First, their method generates a polygonal partition of the area. The partition takes into account the capabilities of each individual vehicle, such as flight endurance and range. Then, the resulting areas are assigned among the UAVs, who could cover them using a zigzag pattern. Each UAV has to compute the sweep direction which minimizes the number of turns needed along a zigzag pattern. Algorithms are developed considering their computational complexity in order to allow near-real time operation. The proposed method is validated through simulation only.

Jimenez et al. (Jimenez et al., 2007) proposed a method for complete CPP based on genetic algorithms. The workspace is considered to be known and planar, populated with polygonal obstacles. The environment is divided in subregions using the trapezoidal cellular decomposition method (Latombe, 1990) and a adjacency graph is obtained. Then a path for each region is created using lawnmower or spiral patterns. Since finding a solution for several subregions is time consuming and computationally costly, a genetic optimization algorithm is used to provide efficient coverage path planning. In this study, the genome is made of two parts: head and body. The head contains the order of the sub-regions as well as starting and finishing points. The body holds the solution for each sub-region. The evaluation function considers the total distance travelled and time. The population is initialized randomly and maintains a constant size throughout the execution of the algorithm. In order to check the viability of this approach the optimal path is tested in a virtual environment.

Mannadiar and Rekleitis (Mannadiar and Rekleitis, 2010) presented a algorithm for the complete coverage of a known 2D environment. The presented algorithm encodes the areas (cells) to be covered as edges of the Reeb graph. It performs an offline analysis of the environment, construction of the Boustrophedon Cellular Decomposition (BCD) and the Reeb Graph (RB), and solves the Chinese Postman Problem for the calculation of the cell ordering. The single cell coverage used in the Boustrophedon Cellular Decomposition algorithm is modified in order to eliminate repeat coverage by splitting selected cells. Cells are covered by using a back and forth motion.

Xu et al. (Xu et al., 2011) extended the previous optimal coverage algorithm (Mannadiar and Rekleitis, 2010) for the general class of non-holonomic robots, particularly UAVs. Previous developments (Mannadiar and Rekleitis, 2010) primarily focused on the offline components of the

algorithm (calculation of the Eulerian circuit), and assumed that the target vehicle could move in all directions independently. The general strategy involves computing a trajectory through a known environment with obstacles that ensures complete coverage of the terrain while minimizing path repetition. The trajectory is then interpreted by a robot controller that accounts for the dynamics of the vehicle. The online motion planner can adjust the coverage footprint width to compensate for the deviations in trajectory. Extensive experimental results in simulation validate the presented system, along with real data from coverage flights using an UAV.

Some interest has also been reported for introducing robotics in agriculture in order to increase the efficiency of the procedures. Oksanen and Visala (Oksanen and Visala, 2009) developed two (greedy) approaches for solving the 2D CPP in the case of agricultural applications with non omnidirectional vehicles. In the first algorithm, offline, the goal is to split a single complex shaped field into convex subfields that are simple to operate with the lawnmower pattern. This algorithm utilizes trapezoidal decomposition followed by a trapezoid merging procedure, and then a search is performed for the best driving direction and selection of subfields. The cost function used in selection considers the normalized area and distance of the block. The second algorithm, online, is also an incremental algorithm, but the path is planned on the basis of the machine's current state and the search is on the next swath instead of the next subfield. The driving is started by following some edge of the field, and after each swath all the possible routes are simulated over a certain prediction horizon and the best of these is applied. This is continued until the whole field is covered by the operation. There are advantages and disadvantages with both algorithms, neither of them solving the coverage path planning problem optimally.

Lee et al. (Lee et al., 2011) presented an online complete CPP and control algorithm for a mobile robot based on a high resolution grid map representation that is capable of generating a very smooth coverage path through Bezier curve approximation. While most approaches try to minimize path overlap, this work focuses on smoothing the coverage path to reduce accelerations and yet to increase the average velocity for faster and more energy efficient coverage. The proposed algorithm adopts a high resolution grid map representation to reduce directional constraints on path generation. For area coverage, they proposed an integrated sequence of spiral path tracking, wall following, and virtual wall path tracking, regarding the covered region as a virtual wall. Wall following is executed by a reactive path planning control process, whereas the spiral (filling) path and virtual wall path are first modeled by their relevant parametric curves and then tracked via dynamic feedback linearisation. For complete coverage, these independent behaviours are linked through a path linking strategy called a coarse-to-fine constrained inverse distance transform (CF-CIDT), which finds the nearest unexplored cell as well as the shortest path to it, simultaneously. The simulation results show that the proposed algorithm generates a smoother, faster and energy efficient coverage path than conventional approaches based on approximate cell decomposition or zigzag motion. However, the increased amount of computation compared to the conventional low resolution grid approaches, naturally arising from using a high resolution map, may cause a problem when applied to a very low powered computing environment.

2.2.2 Planning in 3D environments

More recently, 3D coverage algorithms that use a model of the environment have been presented. Hert et al. (Hert et al., 1996) presented an 3D online CPP algorithm for an AUV moving in an unknown underwater environment. The algorithm was designed with an emphasis in efficiency: guarantee complete area coverage with low overlap. 3D coverage is achieved by applying a 2D terrain-covering algorithm in successive horizontal planes laying at different depths. It operates on the horizontal plane in three steps; zigzagging, inlet covering, and island covering. A robot following this algorithm will zigzag along parallel straight lines to cover a given area. Portions of the area that either would not be covered or would be covered twice are detected by the robot and covered using the same procedure. By covering these small areas (inlets) as soon as they are detected, they are covered in a depth-first order. The spacing of the parallel lines in the zigzag pattern is determined by the robot's camera image width. The robot enters a detected inlet by moving along its boundary and exits by resuming its path as if the diversion inlet did not exist, assuring that the detected inlet is only covered once. However, this algorithm disregards the correlation between successive horizontal planes. Since it does not focus on the efficiency of the coverage result in the new plane, it cannot guarantee the efficiency of the three dimensional terrain covering process.

In order to improve the efficiency of this algorithm, Lee et al. (Lee et al., 2009) developed the artificial island technique that aims at incorporating the prior terrain information into the conventional coverage method. In the sequential process described previously of covering successive horizontal sections of the environment, they acquire information from each covered plane before exploring other plane. The technique involves four steps. First a certain plane is covered by an ordinary planar covering algorithm. Then the safe area is extracted from the plane. Safe areas do not contain obstacles or boundaries. The artificial islands are then by arranging the safe area. The upper plane is then covered by referring to the artificial islands. The procedure is recursively executed until it ends. The main idea behind the artificial island technique is that the AUV covers the area excluding parts of the safe areas. The artificial island technique has the advantage of reducing the covering path length and the time cost for the robot, because it considers the correlation between every plane. Through various simulations, the authors validate the efficiency in terms of the total path length and the running time of the AUV compared to that of the original Hert algorithm.

Jin and Tang (Jin and Tang, 2011) developed a 3D terrain field CPP algorithm that classifies the field terrain according to slope steepness and then applies the most appropriate path planning strategy to each region in terms of minimized headland turning cost, soil erosion cost, and skipped/overlapped area cost. This work addressed four critical tasks: terrain modeling and representation using spline models, coverage cost analysis (headland turning cost, soil erosion cost and cost from the curvature of the path), terrain decomposition and the development of optimized path searching algorithm. The searching algorithm tries to find the best contour curve for each subregion and parallel coverage paths (not rectilinear) are generated by offsetting the found curve toward its two sides until the whole region was completely covered. Compared with the 2D

planning results, the experimental results of 3D coverage path planning showed its superiority in reducing both headland turning cost and soil erosion cost.

Geng et al. (Geng et al., 2014) presented an algorithm for urban surveillance mission planning using UAVs equipped with cameras. The camera, mounted on gimbals fixed to the UAVs, has the capability of panning and tilting. Thus, not only the trajectory of the UAVs, but also the schedule for orienting the camera axis needs to be provided in the mission plan. They solve the problem with a two-stage approach. In the first stage, a set of candidate camera configurations (location + camera axis) is identified. The sum of coverage of these camera configurations forms a super set for covering the target area. The search for optimal camera position is done using Particle Swarm Optimization. When maximum coverage is the objective, the fitness is simply the number of covered sample points. When the minimum exposure is the objective, the fitness is the accumulated amount of exposure to all the radars in the region. In the second stage, the optimal set of camera configurations is constructed from the super set using genetic algorithms. For this optimization problem the flying paths need to be generated for evaluating each candidate set. The authors adopted the genetic algorithm (GA) formulation provided in (Beasley and Chu, 1996) for solving the minimum covering set problem, where each gene represents a camera configuration. The coverage mission is then modeled as a TSP over a graph, where one or several UAVs are required to traverse the defined camera configurations. The effectiveness of the developed methods/algorithms has been demonstrated through simulations.

Hameed (Hameed, 2014) presented two offline approaches to plan 3D field coverage operations for agriculture applications. Energy consumption models are supplied taking into account terrain inclinations in order to provide the optimal driving direction for traversing the parallel tracks and the optimal sequence for handling these tracks under the criterion of minimizing direct energy requirements. The first algorithm is a two step approach where a 2D field representation is used with a GA to find the best possible driving angle minimizing the number of tracks. Then a 3D representation of the environment is used with energy consumption models to obtain the best sequence of tracks. The second approach is a 3D Exhaustive Search among all possible integer values of driving angles. It includes a simulation tool for handling operations with capacity constraints. The author states that the high computational requirements of the latter algorithm prohibits its application as an online tool. The case studies showed a reduction in the energy requirements by taking into account the 3D field terrain.

Lee and Lee (Lee and Lee, 2014) expanded the artificial islands terrain coverage algorithm (Lee et al., 2009) and presented an the hybrid terrain coverage framework which considers various surface conditions in 3D environments. It is designed for underwater terrain exploration in complex environments, with incomplete information, comprised of terrain close to a plane with gradual slopes and terrain with precipitous slopes, such as basins and large vertical sinkholes. This approach incorporates a planar terrain coverage algorithm, a spiral path terrain coverage algorithm, and a hybrid decision module to recognize and select the most suitable technique depending on the sloped surface variations. According to the slope of terrain where the AUV is sensing, an underwater terrain is classified as a gradual or steep terrain. The AUV evaluates which type of terrain is

being sensed by comparing the slope of the sensed region to a threshold value of the slope angle. If the underwater surface is a gradual terrain, it can be approximated by a 2D plane and therefore, the artificial islands terrain coverage algorithm (Lee et al., 2009) can be used. To use the 2D planar coverage algorithm for steep terrain exploration, the 3D space is divided into 2D planes at a uniform interval. Covering every plane and elevating the AUV to the next plane causes unnecessary stopping, turning, and acceleration motion of the AUV. Therefore, the authors argue that a simple boundary following motion using a spiral pattern is sufficient to cover this steep terrain and zigzag motion is not necessary. There is no reason to explore empty space throughout that plane since lower depth planes near the surface had already acquired the relevant information. In this way, the proposed algorithm reduces unnecessary energy consumption. Simulation results show that the proposed algorithm is more efficient than the conventional terrain coverage algorithm (Lee et al., 2009) in terms of the energy consumption of the underwater vehicle.

Dogru and Marques (Dogru and Marques, 2015) presented a method for 2D path planning on 3D terrain. This work studies three related problems. The first problem is derivation of a mathematical energy consumption model for a robot operating on hilly areas. The authors derived a mathematical loss model that takes inclination into account and is based on the copper losses of a DC motor. The second problem is selective measurement and modeling of terrain elevation. They propose sampling the terrain in a spiral pattern acquiring friction and elevation measurements which are then used to model the terrain using Kriging. The final problem is CPP for a hilly and cluttered environment. They assume a lawnmower pattern and try to optimize the sweep angle of the robot and find the order of convex partitions to visit. They use a GA, where the chromosomes represent both the sequence of the partitions, and the orientation of each partition's template. The cost function has two terms: one corresponding to translational motion and taking into account the current slope of the terrain as well as the heading of the robot and other corresponding to rotations. Calculating the cost for a partition takes considerable time so the cost for each partition and its corresponding orientation are stored in a lookup table for future use. Simulation results show that this approach is effective in reducing energy consumption.

Torres et al. (Torres et al., 2016) presented an 2D approach for the coverage path planning problem with an UAV for 3D terrain reconstruction. Their algorithm can deal with both convex and non-convex areas and their aim is to obtain a path that reduces energy consumption, through minimizing the number of turns. The CPP algorithm for single convex polygon coverage involves the calculation of the optimal line sweep direction and the construction of the coverage path using a lawnmower pattern where the rows are perpendicular to the optimal line sweep. They presented a technique that determines the direction in which the path is defined. Initially, measure the length of the lines from every edge of the polygon to the farthest vertex. The line with the shortest distance is the optimal line sweep (the one that will generate less turns in the coverage path). If the polygon to cover is non-convex, coverage is performed as if dealing with multiple convex polygons. However, if the trajectory intersects the concave polygon the best of the following two alternatives is taken: find a non-optimal line sweep and check if the coverage can be done and decompose the polygon as a set of convex subpolygons using any algorithm from literature. The

order for covering all the polygons is determined by minimizing total path length. For a small number of areas, all permutations of polygons and coverage alternatives are evaluated whereas for a higher number, they propose finding the best closed path and forcing some restrictions in the possible polygons orders to reduce number of alternatives.

2.2.3 Planning considering uncertainty

In many situations it is beneficial to consider uncertainty in the navigation data during planning, trying to minimize the vehicle's state uncertainty along the planned paths.

Acar and Choset (Acar and Choset, 2002b) present a planner that does not rely heavily on the absolute position of the mobile robot, but instead they assume that the robot navigates through a free space that has been previously mapped via Morse decomposition and follows the boundary of the cells minimizing the accumulated dead-reckoning error at intermediate critical points (follows their previous work). This could be used as a mobile robot navigation procedure, but only if a cellular decomposition, such as the one described above, is available to the planner.

Some techniques for 2D coverage planning under uncertainty were specifically developed for underwater environments, trying to maximize to quality of bathymetric data acquired during a survey. The actual path followed by the autonomous vehicle can deviate from the planned path due to deficient control technique to follow the path, external disturbances such as currents or sea states that cause deviations may exist or because the location estimate contains uncertainties and errors as a result of noisy sensor data. The benefits of incorporating uncertainty into planning, however, come at the cost of much higher computational complexity.

Hollinger el al. (Hollinger et al., 2013) presented an algorithm for accurate reconstruction of the seafloor that plans the AUV's dives seeking to maximize variance reduction along the planned path while also satisfying a budget constraint on maximum path length. They model the expected accuracy of the map using non-parametric Bayesian Regression in the form of Gaussian Processes. The GP representation provides a measure of variance, which is used as a measure of uncertainty, as well as a mean value of altitude at each point of the bottom grid. Then they propose a greedy method to reduce this uncertainty by calculating efficient dive patterns: greedily select dives that maximize variance reduction until the budget is reached and then run a gradient optimization that perturbs each dive and locks the pattern into a local optimum. They estimate the variance reduction by first calculating the sum of variance in the area viewed by the dive. For overlapping dive patterns, the reduction in variance caused by each subsequent view is approximated using an exponential drop off, which was found to provide sufficient accuracy while remaining computationally tractable. Their experiments indicate a performance gain when compared to standard lawnmower patterns. It was also shown that replanning dives, as new information is acquired and the uncertainty model is updated, provides some improvement.

Houts et al. (Houts et al., 2012) presented an approach for planning terrain-following trajectories that allows an AUV to survey an area in close proximity to rugged terrain. This is useful for missions where AUVs are used to collect a time series of images of the seafloor to monitor it for change. The approach fuses reactive obstacle avoidance with anticipatory information from

the Terrain-Relative Navigation (TRN) system. They use avalilable bathymetry data (i.e. a priori maps) to plan aggressive but feasible paths for a AUV to "fly" through terrain. The implemented safety logic makes decisions based on the FLS and the TRN system. The FLS compares measurements to the map to check for unexpected obstacles. TRN provides both an estimate of the vehicle's position in the map and a measure of the uncertainty of that estimate. The desired primary mode is planning and "flying" aggressive terrain-following trajectories. When confident that the world around the vehicle agrees with the map, it will "fly" an aggressive terrain-following trajectory. So long as the FLS measurements agree with the map and the TRN uncertainty is low, the safety logic controller does not override the depth or altitude commands and the aggressive trajectory will be flown. Otherwise, it reverts to one of the more conservative modes. The approach is demonstrated through simulations using field data from AUV operations in Monterey Bay.

Galceran et al. (Galceran et al., 2013) present a survey path planning technique which takes into account the robot's motion and sensing uncertainty and seeks to minimize this uncertainty along the planned path. They deal with the application constraints of surveying the target area in parallel tracks, at a certain constant altitude from the seafloor, and avoiding turns in order to maximize the quality of the sonar readings. A graph is built, representing the parallel tracks required to cover the target area. Then, a survey path is planned using two steps: first find the best possible order in which to cover the parallel track edges of the graph which minimizes the overall uncertainty along the path; second by inserting crossing track edges in the path found in the first step if, after tracing a parallel track, the uncertainty surpasses a given threshold. The parallel track order is determined by using an algorithm which considers the saliency (Itti et al., 1998) of the terrain, which is computed for every point of an a priori bathymetry of the target area. Then, the robot's position uncertainty along the determined path is estimated using a particle filter with the a priori bathymetry and simulated multibeam sonar measurements. Whenever the uncertainty after a parallel track exceeds a user-provided threshold, a crossing track through a salient area is inserted, seeking to reduce uncertainty. They are interested only in the uncertainty of the robot's belief rather than in the position estimate. Their planning heuristic does not guarantee an optimal path with respect to uncertainty but it tackles the intractability of the planning problem by producing a low uncertainty solution, as demonstrated by their experimental results. On the other hand it can lead to lengthened paths due to the addition of multiple crossing tracks.

Zhu et al. (Zhu et al., 2014) presented map building and path planning algorithms for an AUV navigating in an unknown environment. The work considers the uncertainties of the ultrasonic sensor measurements and makes use of the Dempster-Shafer formulation fusion approach (Pagac et al., 1998) to fuse sensor information and build the map. Whenever the AUV moves, it catches new information about the environment and updates the old map. The task of the map building system is to process the sensor data in order to assess, as accurately as possible, which cells are occupied by obstacles and which cells are empty and thus suitable for AUV navigation. The fundamental concept of the proposed planning approach is to develop a neural network architecture, whose dynamic neural activity landscape represents the dynamically varying environment. The position of the a neuron in the neural network uniquely represents a position in the 2D workspace

of the AUV map. There are only local lateral connections among neurons. Thus, the computational complexity depends linearly on the neural network size. The AUV path is directly planned from the dynamic neural activity landscape and previous AUV position without any prior knowledge of the changing environment.

Kuhlman et al. (Kuhlman et al., 2014) introduced a planning algorithm for computation of coverage plans for persistent monitoring tasks with static obstacles and made a demonstration for USV harbor patrolling. The algorithm modifies an initial informative coverage path towards regions with high information value, while avoiding obstacles and considering environmental effects that cause uncertainty in a vehicle's motion. The authors extended the approach presented by Soltero et al. (Soltero et al., 2012) by developing a waypoint feedback policy that alters the informative coverage plan to minimize expected path traversal cost, accounting for motion uncertainty and presence of obstacles, and maximize information gathered along the path. They formulated a Markov Decision Process problem for each waypoint so the vehicle generates collision free feedback plans to traverse from waypoint to waypoint and can predict expected path traversal costs. The problem is solved using the probabilistic value iteration (LaValle, 2006) and its solution (expected path traversal cost and waypoint-steering feedback plan) is directly integrated into the coverage plan. The single objective function considers the trade-off between the competing objectives of steering waypoints to informative regions versus reducing path length. A direct execution of the waypoint feedback policy results in a rubber-band like contraction of the waypoints wrapping around obstacles, with waypoints being attracted to the informative regions. To avoid local minima, obstacles are ignored during the computation of an initial informative coverage path. The algorithm considers obstacles in later planning stages during which it repairs the portions of the path that are invalidated by the obstacles, and optimizes the path with respect to the environmental effects that cause uncertainty in robot's motion. Simulation results show that the proposed algorithm is robust to motion uncertainties.

2.2.4 Planning mine countermeasures operations

Introducing robotics to mine countermeasures (MCM) operations, or other generic search operations, provides significant improvements in search time and efficiency compared to more traditional methods like using a large manned vessel towing a sensor platform over the search area. The autonomous vehicle is launched and recovered from a surface ship and carries mine reconnaissance sensors that are used to locate and identify objects.

Stack et al. (Stack and Smith, 2003) presented a 2D coverage algorithm for MCM using cell decomposition. They investigate a planning scheme for incomplete coverage. This scheme divides the search area into cells and surveys each cell using a line-sweep pattern with a row spacing that is larger than the sensor footprint, exploring the fact that mines are normally placed in lines. They assume that if mines are evenly spaced, then randomly varying the spacing between each row in the lawn-mowing pattern will decrease the probability of missing an entire mine line. A certain POD is ensured by establishing bounds on row spacing. A perfect POD is assumed for any mine within the sensor footprint. Their coverage scheme aims at minimizing the size of the uncovered regions

while keeping them distributed over the search area. The placement of the rows on the right and left sides of the area is random while the placement of the rows in the middle is a function of the estimated undetected mine locations.

Fang et al. (Fang and Anstee, 2010) developed an algorithm for 2D offline global mission planning, involving the (boustrophedon) decomposition of the surveyable area into subareas using an approximation to the generalised Voronoi diagram, calculation of paths within subareas that allow for incomplete sonar coverage, and connection of subarea paths with transits to obtain a mission plan. They intend to cover a well-known planar seabed using an AUV fitted with a sidescan sonar, maximizing area coverage rate. Paths within each cell were aligned with the edges of the cell produced by the decomposition process. They consider both even and uneven lawn-mowing coverage patterns but the spacing is fixed, proportional to range setting. Since they assume a planar seabed, they cannot consider a complex topography and therefore there was no need to implement variable spacing between the segments.

Williams (Williams, 2010) presented an 2D offline coverage algorithm for MCM that optimizes the spacing between parallel tracks in order to maximize POD, considering seabed type and range. It is assumed that the probability of detecting a mine, if one is present, improves to the maximum of any single view of a given location. They consider a scenario where an area must be surveyed to a certain detection level, regardless of the amount of energy and time required to execute a specific mission plan (they assume that the AUV has sufficient energy). Their track-spacing algorithm consists of an exhaustive greedy search for the best tracks. At each iteration, the gain in POD caused by every possible track is calculated and the track that maximizes it is chosen and added to the set of tracks to traverse. Because of the greedy approach employed, the set of selected tracks is necessarily ordered in terms of decreasing gain in average POD. To further improve these selected tracks a small geographical displacement of each track is considered. This process is iterative and executed until no improvement is observed. The main disadvantage of this greedy algorithm is that solutions are only locally optimal.

Morin et al. (Morin et al., 2013) developed an offline coverage path planning algorithm for AUVs for performing MCM operations. They assume imperfect sensors and that multiple scans of the same area are independent. The POD depends on the seabed type of the surveyed region and on the range of the sensor. They use a cellular decomposition to represent the ocean floor by a grid of uniform square cells. The goal is to plan a path that achieves the minimal required coverage in each cell while minimizing the total travelled distance and the total number of turns. The robot can only move on the grid lines between the cells that represent the environment, therefore limiting the direction of movement of the vehicle to four directions. This is the big disadvantage of their algorithm. It is based on dynamic programming and on a travelling salesman problem reduction. The algorithm is composed of two main phases. In the first phase, a greedy technique constructs a set of disconnected vertical or horizontal segments such that a robot travelling along these segments will achieve the required coverage. In the second phase, they use a TSP reduction to optimally connect the segments obtained in phase 1, obtaining a feasible path of minimal length.

Paull et al. (Paull et al., 2013) presented an approach to 2D seabed coverage for MCM missions

using a SSS. Paths are planned using multi-objective optimization involving information theory and branch entropy based on a hexagonal cell decomposition. Although the proposed method is able to cover target areas with non-convex shapes, obstacles present amidst the workspace are not considered. The backbone of the proposed approach is an objective function that is evaluated over the domain of all possible desired headings. This function represents global utility and takes into account the information gain, the branch entropy and the benefit of maintaining the current heading. The information gain component prioritizes headings that cover the most area in the short term, the branch entropy component prioritizes over headings that will help the agent complete its coverage mission in the longer term and the last component prioritizes over headings closest to the current heading so that obtained SSS data is valid (no distortion). The optimization takes place over heading reference only and it is assumed that desired speed and depth are generated by some other method. In this case speed and altitude reference are held constant. The evaluation of the multi-objective function is done using Interval Programming. Then they use a sonar modelling tool to evaluate sonar performance according to the environment type and distance to target. The main advantages are that the AUV is able to maintain heading for better data mosaicing in the presence of currents or erratic waypoint tracking behaviour and it can adapt to changes in environmental conditions that can be detected in situ. One disadvantage of the method is the need to tune the weights in the objective function. No information is given describing the methodology used to determine the set of weights in the objective function, which is an important process that dictates the quality of the final solution. This method is demonstrated in simulation and experimentally on an AUV conducting MCM operations. The results also show that the information gain approach alone is not sufficient as the mean path lengths are longer then in the cases using lawnmower tracks (common issue in greedy approaches).

2.2.5 Generic survey mission planners

Das et al. (Das et al., 2010) presented a 2D mission planner for robotic surveys in the ocean targeting the detection of harmful algal blooms. They use remote sensing (satellite and radar) to identify relevant bloom patches. After these areas are identified, an appropriate sampling path for the AUV is calculated. They use a lawnmower pattern with constant swath width and assume that the parameters of the bounding box that defines the survey area are predecided. Their goal is to achieve maximum spatio-temporal sampling resolution at the regions of interest, thus maximizing the total signal intensity in that region. To determine the location and orientation for a survey with a known layout and dimension they fit a bounding box to the identified region that has the maximum likelihood of an ongoing bloom. Then a survey path is generated within this area.

Yan et al. (Yan et al., 2012) developed a mission planner for oceanographic surveys which included a fault recovery architecture. The general idea of their algorithm is that some subareas need to be mapped using an AUV by performing a sonar survey following a lawnmower pattern. It maintains depth or altitude during the survey. They adopted a mission planning algorithm (Fang and Anstee, 2010) to guarantee the complete coverage for each subarea, thus inheriting its disadvantages. In order to interconnect all the subareas while minimizing the length of the path,

the authors formulated the problem as a prize winning salesman problem (PWSP) but no further information on the algorithm was provided.

Galceran and Carreras (Galceran and Carreras, 2012) created and algorithm to generate a coverage path over a 2.5D surface on the seafloor using an AUV that navigates at constant-depth. The proposed method tries to minimize the coverage overlapping by segmenting the target surface in regions of similar depth features using K-means clustering algorithm and addressing them as individual coverage path planning problems. Morse-based cellular decomposition is applied to each region. The surface gradient is used to determine the best sweep orientation in each cell, and the inter-lap spacing in the lawnmower-like paths used to cover each cell is maximized on a lap-by-lap basis. The start-to-goal path planner A* (Russell and Norvig, 2003) is then used to search for complete coverage paths concatenating paths computed for each region. The proposal is validated in simulation experiments conducted with a real-world bathymetric dataset that show an increase on path efficiency in comparison with a standard boustrophedon coverage path.

Barrientos et al. (Barrientos et al., 2011) presented a multiple vehicle algorithm for solving the 2.5D area coverage problem for agriculture applications. They developed a procedure that handles the area division using approximate cellular decomposition and assigns areas to the robots. Then the CPP algorithm, based on a wavefront planner (LaValle, 2006) and breadth-first search (BFS), is performed for each subarea. A cost-efficient multiple vehicle CPP algorithm generates a coverage path for each robot, such that the union of all paths is equivalent to the overall coverage and the total coverage time to completion is minimized. Cells containing obstacles were configured as "no fly" zones. They consider as constraints to number of turns, the number of times a cell is covered, coverage time of a single area and total coverage time.

Franco and Buttazzo (Di Franco and Buttazzo, 2015) presented an energy-aware path planning algorithm that computes a path for achieving the full coverage of a given survey area considering the features of the UAV (e.g., the available energy, the weight, the maximum speed) and other mission requirements (e.g., the spacial resolution of the acquired images and coverage). They propose a simple algorithm that optimizes the path by reducing the number of turns in the lawn-mower pattern by setting the scan direction parallel to the longest bounding line. The size of the projected area, needed for track spacing optimization, depends on the altitude of the vehicle and the angle of view of the camera. The paper described a method for deriving an energy model of a specific UAV starting from real measurements. Using such a model, the proposed algorithm computes the speed that minimizes energy consumption for each segment in the path. Since energy is minimized along individually path segments, the resulting solution is suboptimal. Then, a feasibility test is performed to verify whether the energy available on the UAV is sufficient to scan the entire area. If the feasibility test is passed, the remaining energy can be used to increase the spatial resolution of the acquired images. This can be done by iteratively reducing the flight altitude, recomputing the path, and rerunning the feasibility test, until an altitude is found. If the feasibility test is not passed, the path has to be redesigned.

Shnaps and Rimon (Shnaps and Rimon, 2016) presented offline and online algorithms to solve the battery powered coverage problem, where a robot has to cover a planar environment using a

battery of limited capacity. The offline methodology assumes full knowledge of the environment and consists of two interleaved phases: one phase executes full coverage of the environment and the other phase guides the robot back to the charging station when needed. During online coverage, no prior knowledge of the environment is assumed but the robot is able to accumulate information about the obstacles and can return at any time to the charging station. They use an estimated potential function, defined as the minimal travel distance between a point and a starting position only through known environment, which is updated during task execution. The experiments were successful but there were difficulties with the localization system and it needed calibration during the coverage process.

2.2.6 Summary and outlook

Several algorithms were presented tackling different derivations of the CPP problem. Some mission planners were also discussed, addressing the maximization of mission performance while considering the energy expenditure.

Over a long time period, the benefits of adaptive autonomous search are overcome by performance of standard preplanned search paths (Baylog and Wettergren, 2014). If the goal of the mission is to find an object as soon as possible (as in search and rescue operations), it would be advisable for the vehicle to react to new and relevant information immediately. However, if the goal is to find all the objects in the area over a fixed time period then it is more beneficial to follow the preplanned paths for the duration of the search.

Since many information is available a priori, a path should be precomputed offline. The path should be modified only if the newly acquired information while the mission is being executed is found to be significantly different. The amount of data that needs to be searched for true 3D path planning can make the search task infeasible if relying on the limited processing power available in the onboard computational resources. Thus, there should be a realistic balance between the reduction of the path search space and search performance considering the available computational resources. In this work, an interesting solution for offline and online mission planning that addresses these problems, while accounting for uncertainty in navigation, is presented.

2.3 Learning in Robotics

The ability of an agent to survive in a dynamic environment depends essentially on the number of experiences that he has learned and retained in its memory (Dempsey et al., 2003), which controls its behaviour (Noice and Noice, 2006) and may even be responsible for its survival (Glickman et al., 2005). The way an agent learns the behaviour, organizes it, and stores it in its memory could be a measure of the level of the intelligence of the agent (Craik and Lockhart, 1972). The ability to learn has been developed using several types of well known learning algorithms such as reinforcement learning and case-based learning. A self-adapting, self-organizing decision making system is one of the possible ways to tackle and manage higher complexity present in dynamic environments.

2.3.1 Evolutionary algorithms

Evolutionary computation originated in the late 1950's, after computers began to be commercialized. Bremermann (Bremermann, 1962), Box (Box, 1957) and Friedberg (Friedberg, 1958)(Friedberg et al., 1959) researched into areas that evolved into currently available techniques known today has Evolutionary Algorithms (EA). Developments in the area were slow due to the fact that the computer systems available at the time were rare, expensive and lacked processing power, which was normal since they were on its early days. Then the area remained relatively unknown to the scientific community for decades. In the 1970's, the work presented by Rechenberg (Rechenberg, 1973), Schwefel (Schwefel, 1975) and Holland (Holland, 1975) marked the beginning of an exponential increase on the number of publications in the field. Modern EA techniques often require very large amount computer resources. The continuous improvement in hardware technology has allowed the use of EA techniques with higher complexity, providing means for addressing real world problems that traditional algorithms were unable to conquer. This is the reason why the field of evolutionary computation is one of the fastest growing areas of computer science and engineering.

Evolutionary algorithms are a flexible and adaptable set of techniques used for learning and problem solving, well suited for solving complex optimization problems. There are several variants tackling different issues, but they are based on the principles of natural evolution as presented by Darwin (Darwin, 1859). The techniques computationally simulate the natural evolutionary process by incorporating Darwin's concept of "survival of the fittest". It states that the fitter an individual is, the greater is the chance for that individual to have a longer life and consequently having more chances of generating offspring. The offspring will inherit the best parental information. Fgoel Evolution can be interpreted as adding new individuals to a population, sharing information of the fittest individuals. The nondeterministic nature of the reproduction process leads to a permanent creation of new information and therefore to the generation of diversified offspring.

Basically, an EA consists of a population of candidate solutions, known as individuals, manipulated by a set of operators, called variation operators, and evaluated by a given fitness function. The fitness function determines how good an individual is and assigns a corresponding fitness value to the individual. The calculation of the fitness values involves in some way the evaluation of the objective function defined on the problem being solved. An individual's associated relative fitness dictates if he will survive into the next generation. The three major evolutionary operators associated with EAs are selection, recombination and mutation. After the population members are evaluated, the selection operator chooses fittest individuals with a larger probability to fill an intermediate mating pool. The mating pool is an abstract container that holds the selected individuals. For this purpose, several stochastic selection operators exist in the literature. The main idea of the selection operator is to give preference to fitter individuals by selecting them for reproduction. This allows them to pass their best attributes to the next generation. It also excludes the entrance of worst fit individuals into the matting pool, denying them a chance to reproduce. The varia-

tion operators act on the individuals in the population with the goal of generating fitter offspring. The purpose of the recombination operator is to pick two or more (parent) individuals randomly from the mating pool and create one or more individuals by exchanging information among them. The recombination operator is applied with a recombination probability, indicating the proportion of population members participating in the operation. In the context of real-parameter optimization having n real-valued variables and involving a recombination with two parent solutions, each variable may be crossed at a time. The offspring created by the recombination operator are then perturbed in its vicinity by a mutation operator. Mutation is a operator used to maintain diversity from one generation of a population to the next. It alters one ore more attributes of an individual from its initial state. With those attributes, the new individual may become fitter than its parent. Mutation is an important part of the evolutionary search as it is a mechanism designed to prevent the population from falling in any local optima, stopping the evolution. Usually it is associated with an mutation probability defined by the user.

Evolutionary algorithms have two conflicting goals: exploration and exploitation. One goal is only achieved at the expense of the other. Choosing a adequate EA selection pressure parameter is important because it is the mechanism that controls what type of search will be performed.

Finally, an EA execution stops when one or more pre-specified termination criteria are met. Often, a predetermined number of generations is used as a termination criterion. Another option, depending on the type of problem being studied, is to terminate the execution of the algorithm as soon as a solution with a predefined goal or a target solution is found. One can also use a termination criterion based on the statistics of the current population and that of the previous population to determine the rate of convergence.

Although these algorithms are somewhat flexible, careful consideration must be given when mapping from problem to algorithm domains. The performance of the algorithm can be severely affected by an inappropriate representation of the individuals or an inappropriate definition of the variation operators, preventing solutions from being evolved. Although there is no unique combination ensuring good performance, choosing prudently may result in more effective and efficient implementations. Unfortunately, sometimes it is very difficult to determine exactly what representation scheme should be used.

EAs have been employed in the past to solve the CPP problem successfully, as discussed in section 2.2 (Beasley and Chu, 1996, Geng et al., 2014, Jimenez et al., 2007).

2.3.2 Case-based reasoning

Case-based reasoning (CBR) solves a new problem by adapting solutions of similar problems solved in the past to the assumptions of this new problem. A case usually consists of the description of the problem, the solution that was used to solve that problem and its outcome. Essentially, the CBR methodology considers that similar problems have similar solutions. When a new problem occurs, the most similar problem is searched in the casebase. If the old problem and new problem are somewhat different, the solution of the old problem probably needs to be modified to fit to the new conditions. The modified solution is then used to solve the problem at hand. The

new problem together with the new solution and the outcome of that solution makes a new case is stored for later use. Thus in CBR, learning and adaptation are performed by gathering cases. Benefits resulting from the use of CBR will be most significant in applications where robots navigate in less variable environment and often perform similar repetitive tasks. Storage of previous experience for future reasoning leads to learning from past actions. This leads to the problem of managing the required storage space. However, considering the timely responses needed for online replanning of routes during a mission, the extra memory requirements are admissible.

Classical algorithmic approaches like A* have a major drawback that even when the same problem is solved repeatedly, the time and effort needed to compute the solution does not improve. In other words, these methods do not use its own past solutions and intermediate results for subsequent problems. The advantage of CBR compared to black-box techniques, such as neural networks, is that the cases in the casebase are explicitly stored. The operator understands what the robot has learned and why it comes to one or another conclusion.

Aamodt and Plaza (Aamodt and Plaza, 1994) presented a survey of the fundamental notions related to CBR and descriptions of the main approaches on this field. A pure CBR system starts with a initial casebase. To be able to solve all problems it is necessary that the cases in the casebase cover the whole solution space (Smyth and Keane, 1995). The methodology of a CBR system is described by what is called a CBR cycle (4R cycle), which contains the following four steps – retrieve, reuse, revise and retain. These steps represent a simplified description of the CBR system function: retrieving the most similar case or cases, reusing information and knowledge from the retrieved case to solve a new problem, adapting the selected solution and retaining parts of this experience to be used for future problem solving. Selecting the appropriate strategy for maintenance of the case base can increase the efficiency of the CBR system.

In path planning, it is possible, through the casebase, to reuse previously found paths or their parts. Instead of searching for a new path, it could be sufficient to find the most similar paths or their parts and adapt them to be suitable for the current problem.

2.3.2.1 Path planning with case-based reasoning in static environments

Vasudevan and Ganesan (Vasudevan and Ganesan, 1994) present a CBR scheme for path planning in AUV missions. An annotated map database is employed to model the environment and routes which are used in earlier missions are represented as objects in the map. When a new route needs to be planned, the path planner retrieves a matching route from the database and modifies it to suit to the current situation. Whenever a matching route is not found in the casebase, a new route is synthesized based on past cases that describe similar navigational environments. More specifically, it generates a new route by attaching new segments to old ones as well as joining path segments belonging to different routes. This ability to switch from plan repair to plan synthesis helps to reduce the failure rate and excessive tweaking of old cases. The authors report that the time taken to plan routes is also considerably lower than similar work reported. The advantage of reuse of past navigational paths would be more dominant when planning longer routes.

Branting and Aha (Branting and Aha, 1995) developed an approach for using abstraction in CBR named stratified CBR. Underlying this approach is the observation that hierarchical problem solving gives rise to solutions at multiple levels of abstraction. If the solutions produced by hierarchical problem solving at every level of abstraction are retained, then the more abstract solutions can assist in indexing, matching, and adapting less abstract solutions. This approach is exemplified on a path planning task which involves finding an optimal or near-optimal path between a given pair of start and goal positions through a field containing obstacles. The stratified CBR approach to this problem is to retain and reuse the solution paths found at every level of abstraction in hierarchical search. In this case, the levels of abstraction correspond to different grid resolutions. Retaining abstract solutions decreases the cost of finding the most similar stored solution path because the distance from the new start and goal to these paths can be accurately measured at the highest levels of abstraction, where search is much less expensive (since the fields are smaller in size). The results show that it outperforms the A* and other non-CBR algorithms.

A CBR approach for a continuous environment that uses parts of paths was described by Haigh and Shewchuk (Haigh and Shewchuk, 1994). They propose a case graph constituted by linear segments of paths with segments intersecting only at their end points. When a plan is generated, the solution is either stored as a case, or broken into several pieces and stored as a series of cases. If any of these segments are needed for reuse, a similarity metric will estimate the best place within each segment to start using the case. A similarity metric is used to identify relevant cases by estimating the similarity of cases to the problem at hand, taking into account the relative desirability of different cases.

2.3.2.2 Path planning with case-based reasoning in dynamic environments

Maarja (Kruusmaa, 2003) presents a self-adapting approach to global level path planning in dynamic environments. The aim of this work is to minimize risk and delays in possible applications of mobile robots. The approach is an hybrid system that uses CBR as well as grid-based maps for decision-making. Map-based path planning suggests new solutions. The robot uses CBR to remember the paths it has followed and choosing the best paths according to its traversability. To characterize the traversability of every particular path they define a cost function where the parameters are the length of the path and the risk (or difficulty) of following the path, which is calculated by gathering information during the path following process. By remembering the past cases, the robot learns about the environment and uses this information to solve new problems. Forgetting strategies are discussed and evaluated.

Likhachev et al. (Likhachev et al., 2002) presented an approach to learning an optimal behavioural parametrization in the framework of a CBR methodology for autonomous navigation tasks. The learning process can either be a separate training process or be part of the mission execution. In either case, the robot learns an optimal parametrization of its behaviour for different environments it encounters. The idea behind this work is to fully automate the creation of a library of cases, so that the CBR unit creates and optimizes cases in the library as the result of an automatic experimental procedure. At the training stage the system starts off with a completely empty

library. As training proceeds new cases are created whenever there are no close enough matches already present in the library. As the size of the library is limited (for this work a limit of 10 cases was used), if the library is already full then the most similar case is adapted and applied. Whenever a case is selected for the application, it goes through an adaptation step that, based on the previous performance of the case, adapts the case in the direction that resulted in the performance improvement. The case is then applied and its performance is re-evaluated and stored for being use by the adaptation routine the next time the case is selected. Once the training is over and a robot exhibits a good performance in the training session, the library is "frozen" and can be used in real missions. It is important to note, however, that the training does not need to be separated from actual mission execution. That is, an alternative to the above procedure is to have the robot learning the cases as it executes missions. In this case, even though the performance in the first robot missions would be far from optimal, as more and more missions are executed the robot's performance would consistently improve. This system was able to outperform the behavioural system without learning, in which the parameters are held constant throughout the environment. However, one problem that arose is that the case library filled up relatively quickly. Once the library was full, the best matching case to the current environment was adapted, even if it was drastically different. The adaptation changes both the input and output values of the case, and if this adaptation is done on a case that is tuned to an entirely different situation, degraded performance can result. Kira and Arkin (Kira and Arkin, 2004) present a mechanism by which such cases are removed when the case library is full and a new situation is encountered. The general hypothesis is that a forgetting mechanism, where cases are deleted based on specific strategies, can improve the performance of a CBR system. Various strategies are proposed, including metrics such as how frequently a case is used. Instead of forgetting cases by restricting the size of the case library and deleting cases when the case library is full, the authors perform this action periodically. Experimental results show that such mechanisms can increase the performance of the system significantly and allow it to essentially forget old environments in which it was trained in favor of new environments it is currently encountering.

2.3.3 Reinforcement learning

Reinforcement learning (RL) has attracted increasing interest in the machine learning and artificial intelligence communities and is well-suited for use on mobile robots. It assumes that the environment can be described by a set of states and that the agent (the robot in this case) can choose one of a fixed number of actions (or behaviours). At each time step, the agent observes the state of the environment and chooses an action to take. After taking the action, the agent is given a reward reflecting how good that action was, and observes the new state of the environment. The most important difference between RL and other learning methods is that there is no representation of input/output pairs. Instead, after choosing an action the agent is told the immediate reward and the subsequent state, but is not told which action would have been in its best long-term interests. It is necessary for the agent to gather useful experience about the possible system states, actions, transitions and rewards actively to act optimally.

The reinforcement learning paradigm has been successfully implemented for many well-defined, low dimensional and discrete problems (Sutton and Barto, 1998, Thomaz and Breazeal, 2008) and has also yielded a variety of applications in rather complex domains in the last decade. RL seems to be a natural choice for learning control policies on mobile robots (Bakker et al., 2005, Kohl and Stone, 2004, Riedmiller et al., 2000, Smart and Kaelbling, 2002, Stephan et al., 2000, Wiering et al., 2000). Problems in robotics are often best represented with high-dimensional, continuous states and actions. It is often unrealistic to assume that the true state is completely observable and without noise. The learning system will not be able to know precisely in which state it is and even vastly different states might look very similar. Therefore, several problems have been encountered when trying to scale up these methods to high-dimensional, continuous control problems, as typically faced in the domain of robotic systems. Some RL approaches often consider state dimensionality reduction techniques but these are known to limit the dynamic capabilities of the robot (Peters et al., 2011, Schaal et al., 2002).

2.3.4 Artificial neural networks

An artificial neural network (ANN) is a model inspired by the way the human brain processes information (Hebb, 1949, McCulloch and Pitts, 1943). It provides a practical method for learning real valued and discrete-valued functions from examples. A great deal of literature is available explaining the basic construction and similarities to biological neurons. The performance of the neural network is closely related to the efficiency of the training process and the quality of the training data. It also depends on the function being modelled, the chosen network topology and other parameters such as the number of nodes, initial weights and biases, learning rate and activation functions. Too many parameters lead to poor generalization (overfitting), and too few parameters result in inadequate learning (underfitting) (Duda et al., 1973).

Building a ANN involves the solution of two optimization problems: structural optimization and parametric optimization. Structural optimization is the first step that tries to find an optimum system structure. After the system structure is determined, parametric optimization is applied to find the optimum system parameters. In other words, one needs to select a lot of parameters, such as the number of hidden layers, the number of units in each hidden layer, the type of learning rule, the transfer function and the learning parameters.

When training a ANN, input samples are fed to the network and the output is compared to the target values to calculate the error, which is corrected in the next iteration by adjusting the synaptic weights. Several training algorithms have been designed, but the most commonly used is the Levenberg–Marquardt (LM) back-propagation algorithm. A stopping criterion indicates when to stop the training process. It can be a predefined limit of absolute error, mean square error (MSE) or just the maximum number of training epochs.

It is important to develop an algorithm that adjusts the training process to the performance of the neural network in an changing environment. Building a ANN with adequate architecture, training parameters and enough data, makes it possible to achieve a sufficiently small training error. If the data used for training is not representative of the decision space, then significant estimation

errors may take place. Another difficulties are how to improve the generalization capability of the network and manage a continuously growing training dataset without running out of memory or increasing the processing time beyond given limits.

ANN learning has been successfully applied to problems in the robotics domain such as navigation (Pomerleau, 1991), control (Miller et al., 1990), path planning (Glasius et al., 1995, Yang and Luo, 2004, Yang and Meng, 2000), and obstacle avoidance (Beom and Cho, 1992, Glasius et al., 1995).

2.3.5 Summary and outlook

Here, we described how the ability to learn and retain knowledge can be implemented and several application scenarios were detailed. In particular, for realtime applications such as path planning for mobile robots, the strategy of choosing, modifying, and adapting a candidate plan from the plan repository should be faster than building a plan starting from scratch. For AUVs in particular, different missions would share many common features and hence an adaptive planning scheme including CBR is very attractive. As an example, in most underwater missions the probability of travelling between the same start and destination points is quite high. Often, a familiar solution that is certain is preferred over something new and presumably better.

However, most of the machine learning algorithms require a training phase whenever new information about the environment is available, which makes online adaptation difficult. Hence, learning in dynamic environments is complex for most machine learning methods. Another common problem is that, in general, machine learning techniques are data oriented, this is, they model the relationships contained in the training data set. Hence, if the training data set is not a representative selection from the problem domain, the resulting model may not be a good approximation of the real process.

2.4 Measures of mission performance

In order to assess the effectiveness of a search operation, the detection performance that should be achieved in a specific mission needs to be estimated.

2.4.1 Lateral range

The concept of lateral range curve (LRC) was introduced by Koopman (Koopman, 1946). Imagine a searcher following an infinitely long, straight path, searching on either side of that path. Lateral range refers to the perpendicular distance an object is to the searcher's path. It represents the distance from the searcher to the object at the closest point of approach (CPA). A LRC is a plot of the probability of detecting an object on a single pass as a function of the object's lateral range from the searcher's path. That searcher's LRC $p(x)$ is the probability of detecting a stationary object that is at its closest exactly a distance x from the searcher's path. Note that this is not

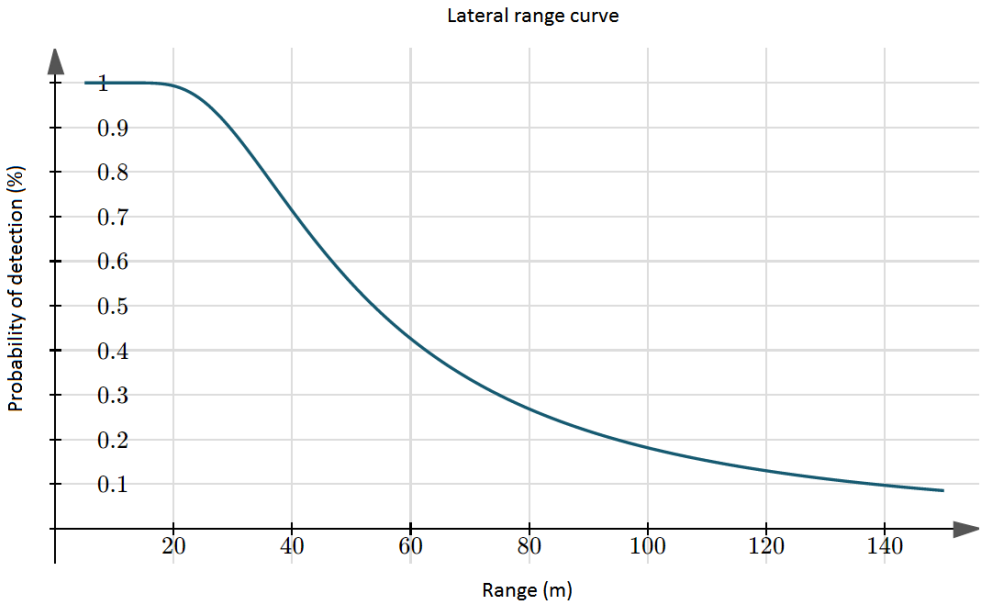


Figure 2.5: Lateral range curve.

the usual POD, which is the probability of detecting an object that is within a distance x of the searcher's path. Figure 2.5 shows an example of a lateral range curve.

Theoretical lateral range curve

In order to calculate a map of POD in the operating area, the POD needs to be calculated at any (x,y) point. Sonar equations are primary analysis tools for studying and predicting sonar performance. Prior (Prior, 2006) presented a simple sonar model and derived a formula for calculating the POD from the beam origin on the sonar, considering sonar and environment properties. Bays (Bays, 2012) derived a model for the estimation of the combined POD relating the probability of detecting a target with a given sonar to the relative angle between the vehicle and the target. The expected probability of the sensor detecting a target at point (x,y) is then:

$$P_s(x,y,S) = 1 - \frac{1}{\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} M(x,y,\phi,S)d\phi = 1 - \frac{1}{\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \prod_{j=1}^{M}\prod_{k=1}^{K^j}[1 - P_r(x,y,s^j)P_a(\phi,\theta_j)d\phi] \tag{2.2}$$

where P_r is the POD given by the sonar model, P_a the probability of detection as a function of relative angle between the vehicle and the target, ϕ is the angle between the axis of symmetry of the object and the ground, θ is the grazing angle (the angle at which the sonar pulse is launched and propagates across the seafloor) and s represents a given sonar swath.

Experimental lateral range curve

Target detection is influenced by many factors which may make the task of building an accurate model of the instantaneous detection unreasonable. The amount of data required to extrapolate a LRC directly is much less than that required to develop an instantaneous detection model. The LRC is derived experimentally by moving a sensor through an area, where objects are randomly placed (the positions are known), using parallel straight-line search transects and then testing the detection system. In order to generate accurate results, the experiment should involve the use of various search objects under very specific conditions. Development of LRCs requires a large sample set with many detection opportunities, where the object positions are known, in order to achieve a more statistically valid curve. Each point on the LRC represents the probability that a target will be detected from a specific closest point of approach range. Derivation of the POD versus lateral range data requires that the search results be processed to identify the range of every detection opportunity for each search object on each search transect. The probability of detection is estimated empirically by:

$$P_D = \frac{number\ of\ detections}{total\ number\ of\ opportunities} \tag{2.3}$$

For each search object type, detection opportunities are grouped by range into bins of different sizes with more bins representing the closer ranges where most detections take place. Then a probability of detection is calculated for each bin with a given percentage confidence limit on the estimation of the proportion of detections to detection opportunities. A model can then be developed using regression techniques such as logistic regression analysis. Model validity can be assessed by feeding random data from the previous surveys, plotting the resulting range curve and analyzing how the modeled curve fits the binned data.

2.4.2 Probability of detection

POD is an estimate of how likely it will be for a search performed in a given area to find an object, assuming it is there. It is a conditional probability since it is assumed that the object is in the area searched. POD measures sensor effectiveness, accuracy, and quality for the search task. It has become the de facto measurement used in search and rescue theory. Unfortunately, even a thorough search of an area does not guarantee finding what was lost. It depends on three things:

- The "detectability index" (effective sweep width) for the combination of search object, search environment, and sensor present in the search task;
- The amount of effort spent in searching the area;
- The size of the area searched.

It is known that an overlap in area coverage can improve the detection performance. The redundant information that is gathered helps to reduce ambiguity in a noisy environment. Also, multiple

surveys from different perspectives reduce the probability that an object is blocked from view. In this sense, another important measure of performance is presented which is the cumulative POD. After covering a given area multiple times, the probability of detecting the search object, if it was present, should be increased as compared to having searched the area only once. However, searching the same area twice does not double the chances of detecting it.

When it is assumed that multiple searches are executed independently of each other, the combined POD is given by:

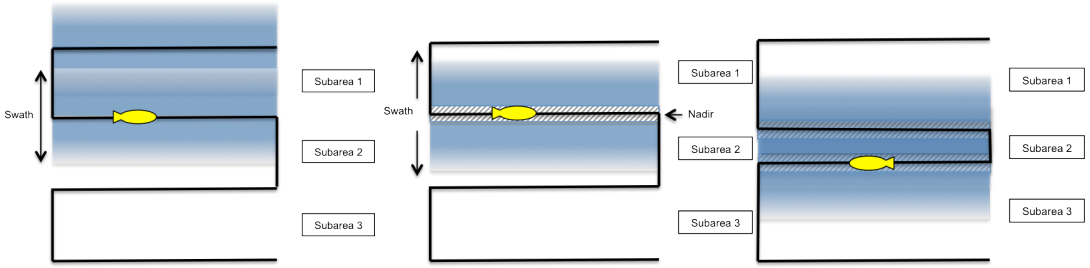
$$PODcum = 1 - \prod_{i=1}^{n}(1 - POD_i) \tag{2.4}$$

The degree of correlation between the information gathered from two searches over the same area needs to have a direct impact upon the manner in which detection probabilities are combined. Here, three different scenarios are considered: no correlation, complete correlation or an indeterminate amount of correlation. When there is no correlation between the two searches, the formula presented above to calculate the cumulative POD is used. If assuming complete correlation, the combined POD is simply the higher of the two probabilities. When there is an indeterminate amount of correlation an accepted practice (Nash et al., 1982, Rollins, 1970) is to average the probabilities obtained assuming complete independence. Modern research involving the calculation of the cumulative POD in different case studies assume complete independence, but there are some problems with this approach (Carnes, 1993). If additional surveys are performed using the exact same path in the same conditions (same sensors, detection algorithms, environmental conditions), the same information would be acquired (same path and conditions guarantees same perspective of the object) in each survey so the several PODs should not be combined since there is no information gain. Still, combining the probabilities should be a close approximation to reality since it is very difficult to replicate the exact same conditions in different experiments. The choice of methodology to be used to combine information from different surveys should also take into account the specific nature of the problem. Minehunting operations involve a danger or risk factor that makes the cost of overestimating the POD much higher than the cost of underestimating it. Ideally, a methodology that produces the most reliable estimates is desired but, when in doubt, it might be beneficial to opt for a conservative estimate. Conservative estimates are obtained by assuming the existence of complete correlation, thus by choosing the higher of the individual estimates.

Care should be taken during mission planning so that the information gain is maximized in those situations where overlapping in coverage is required. Some measures that can be applied are using multiple vehicles with different payload sensors or perform surveys in different directions.

2.5 Covering with different types of sensors

During an operation, the AUV will gather information on the environment and further analysis of that information will allow the detection of existent objects in the area. As the success of the mission relies on the quality of the acquired data, they have to be planned considering the



(a) Coverage with a multibeam sonar (b) Coverage with a sidescan sonar (c) Uneven lawn-mowing technique used for covering the nadir gaps

Figure 2.6: Different methodologies used in coverage problems.

technological constraints of modern AUV systems. Now, a brief description of the characteristics of conventional sidescan sonars and multibeam echosounders is presented to understand how they can be used for search operations.

Sidescan sonars are widely used to produce acoustic images of the seafloor with high resolution for a wide variety of purposes including detection and identification of underwater objects and seafloor features. The operating principle of the sidescan sonar is based on forming an acoustic image by moving the sonar forward and compiling the sonar response from successive beams. It uses one transducer on each side to emit acoustic beams down towards the seafloor, across a wide angle perpendicular to the path of the sensor through the water, and receives the echoes of those beams. They have several advantages, such as being available at a relatively low cost, being easily deployed on AUVs and their ability to produce acoustic images of the seafloor with high resolution making it capable of illuminating small targets such as the typical mines. All sidescan sonars suffer from an inability to illuminate targets within the nadir gap area. This is the part of the seafloor directly below the transducers that, because of the geometry of the sonar configuration, is under sampled. This under sampling exists on both sides of the sonar and extends to a distance close to the altitude of the sonar.

The multibeam bathymetric sonar uses two or more perpendicular transducer arrays to transmit and receive the beams. Each received beam allows the estimation of the range (calculated from the time delay) to the seafloor. The range estimate in each beam and the angle of the received beam gives the relative height of the seafloor relative to the vehicle. By combining data from consecutive pings, a 3D map of the seafloor can be generated. One disadvantage of this type of sonar is the decreasing angular resolution as a function of distance from the nadir resulting in decreased resolution in the outer swath.

The most relevant feature to consider in mission planning is the existence of the nadir gap in sonar coverage, as illustrated in figure 2.6. If it exists then a different technique should be used to optimize the coverage plan. The traditional approach to compensate for this feature is to partially overlap pairs of swaths so their nadir gaps are covered, also known as uneven lawn-mowing.

Chapter 3

Problem statement

The principal problem under study in this thesis is how to design and implement a more flexible 3D path planning algorithm that enables an AUV to efficiently cover the bottom of a submerged area with no missed areas and with a specified minimum POD. The planner should identify a set of parallel tracks, representing sonar swaths, that maximize the estimated performance of a search operation, using the available knowledge and resources. Searching for a path requires the consideration of the environment characteristics (terrain topography), available resources (characteristics of the onboard sensors, available battery power), maximum time available for the mission and vehicle kinematic constraints.

A discrete search modeling paradigm is considered, where the objects of search are stationary and the region to be searched W is partitioned into a collection of N small cells c_i that are mutually exclusive and exhaustive over the region. Our concern is whether the cells are occupied by objects of interest. Cell occupation is not restricted to imply that only one object is present. However, cell sizes are selected such that the likelihood of cells containing more than one object is remote. This facilitates the development of the detection likelihood. The most important aspects to consider when constructing a cell-based paradigm is the cell size and the number of cells to include. The size of the cells is considered to be sufficiently small such that the coverage can be treated as uniform over the cell. But this may become problematic as the greater the number and the smaller the cell size is, then the greater will be the computational demand.

To evaluate the search performance considering the characteristics of the coverage sensors presented in section 2.5, probability calculations are performed over individual cells and the result aggregated over the covered cells according to a search plan. When covering a given area multiple times by navigating through different search tracks, a conservative approach is chosen assuming the existence of complete correlation between the observations. Thus, the higher of the individual probability estimates for the cells in that area is selected.

Obviously the vehicle is confined to the workspace defined by the latitude and longitude of the survey area, the bottom floor and the sea surface. It is assumed that the environment where the AUV navigates, in 3D space, is partially known. Although mapping data of the seafloor is given in advance, it is likely incomplete or inaccurate and there may exist unknown obstacles in

the area. Therefore, knowledge about the environment must be incrementally acquired by using sensors onboard the AUV.

The mission is generally considered complete either when a proportion of cells have been covered to a specified level n_{thresh}. This leads to the formal statement of the mission planning problem.

3.1 Decision variables

Mission path

The path is represented by a set of parallel straight-line transects also referred as swaths:

$$P = \{s_1, s_2, ..., s_n\} \tag{3.1}$$

The swaths represent the data acquired by the sonar while traversing the useful parts of the trajectory. When using the typical lawn-mowing coverage pattern, these swaths are parallel to each other.

Vehicle velocity

The AUV will follow the specified path with a constant forward velocity relative to water equal to v_{water}.

Vehicle orientation

The AUV's orientation is defined as:

$$O = \{o_1, o_2, ..., o_m\} \tag{3.2}$$

where o_i denotes the orientation to be taken, in terms of the three Euler angles roll, pitch, yaw at a specific position in space while the vehicle is following the calculated path. The points in space where the orientation changes are independent of the path nodes.

3.2 Objectives

Maximize probability of detection

One of the goals of the planner is to maximize the search effectiveness, which in this case is represented by the average probability of successfully detecting the target object on the survey area:

$$max \sum \overline{POD_s}(S,P) \tag{3.3}$$

Constraint	Type
Battery capacity (Wh)	vehicle
Percentage of energy available (%)	vehicle
Velocity relative to water (m/s)	vehicle
Maximum steering angle (^{o})	vehicle
Sonar maximum range (m)	sensor
Sonar vertical beam angle (^{o})	sensor
Maximum operating time (h)	mission
Minimum POD (%)	mission

Table 3.1: Design constraints of our optimization problem

Minimize energy consumption

The amount of energy consumed by the vehicle depends on its velocity, orientation and the actual path. Both the energy consumed by the motors and the payload should be considered.

$$min\ E(P, v_{water}) \tag{3.4}$$

Minimize time to complete the mission

The time required to execute the mission depends only on the path and the component of velocity that is parallel to the track direction.

$$min\ T(P, v_{earth}) \tag{3.5}$$

3.3 Constraints

The bounds on the optimizing parameters are presented in table 3.1.

Chapter 4

System model and simulation

4.1 MARES AUV

4.1.1 Platform

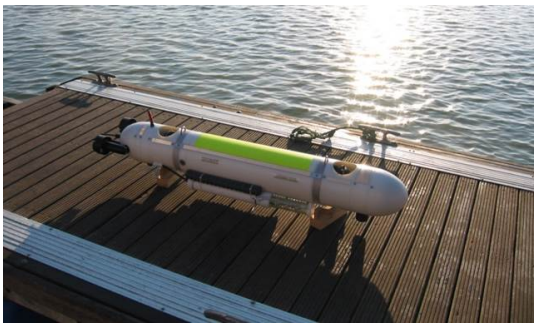
The MARES (Modular Autonomous Robot for Environment Sampling) AUV, shown in figure 4.1, is a reconfigurable AUV designed to operate at depths up to 100 m. It features a plastic hull with a dry mid body (for electronics and batteries) and additional rings to accommodate sensors and actuators. The overall dimensions of the AUV are 1.7 m length, a diameter of 20 cm and weight around 40 kg, the actual weight depending on the particular vehicle configuration and payload. Its modular structure simplifies the system's development (the case of adding sensors, for example) and its compact dimensions allow an easy deployment by trained operators without using cranes or a specific vessel. It is propelled by two horizontal thrusters located at the rear and two vertical thrusters, one in the front and the other in the rear. This configuration allows for small operational speeds and high manoeuvrability, including pure vertical motions. The vehicle can be programmed to follow predefined trajectories while collecting relevant data using the onboard sensors.

Payload

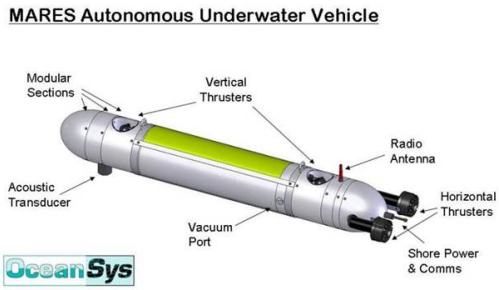
The AUV is equipped with a range of sensors:

- The Imagenex Sportscan side-scan sonar to obtain acoustic images of the seabed;

Figure 4.1: MARES AUV ready for a search mission.



(a) MARES AUV configured for a rapid environmental assessment mission.



(b) Vehicle schematic.

Figure 4.2: MARES AUV in its smallest configuration.

- Keller 36 X W pressure sensor to calculate depth;
- Imagenex 864 altimeter to measure distance to seabed;
- Xsens MTI IMU for inertial navigation;
- Amaryllo Roots GPS receiver for absolute positioning information;
- An acoustic transducer and an electronic system that allows for long baseline navigation, measuring ranges to an acoustic beacon placed on a surface buoy;
- Sea-Bird Electronics 49 FastCAT CTD to measure conductivity, temperature and depth.

Since different types of missions require different sensors, the vehicle physical shape may vary according to its configuration.

Embedded system

The core module is a Diamond Pegasus PC104 Single Board Computer with 512MB RAM for navigation, control and data logging. The onboard software was developed in C++, runs on a Linux kernel and is composed by a set of independent processes.

For heavy data processing a separate (low cost) computer was chosen: the faster, octa-core HardKernel ODROID XU4 with 2GB RAM.

4.1.2 Additional requirements

Reliable estimation of the trajectory is essential to the navigation of autonomous vehicles. In order to function in unstructured, unknown, or dynamic environments, the vehicle must be able to estimate its current state and perceive its surroundings, allowing the generation of actions that are suitable for that environment and for the goals of its mission. The planning, control and navigation systems are critical parts in an autonomous system as each complements the other and

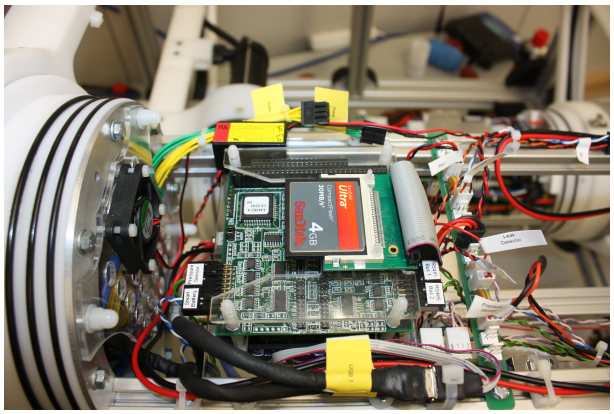


Figure 4.3: MARES processing units.

compensates for the other's deficiencies. For increased autonomy and safety of the system, an estimate has to be provided in real-time with minimal latency. It must also be able to deal with uncertainty and incomplete knowledge of the environment and of the effects of its own actions. A variety of methods have been developed to tackle this problem. The most common sensors are the IMU, that have the advantage of high sampling frequency and relatively small latency, but only capable of providing relative motion and are subject to significant drift. These are often complemented by GPSs which have the advantage of providing a global position at a relatively high frame rate, but with lower accuracy and precision. Since it increases the autonomy of our vehicle, is easy to deploy and to manage and is widely adopted for coverage missions, it was decided that this would be an important feature for our autonomous vehicle to have and therefore will implement it.

One of the variables that is being optimized in this study is the energy consumption. This is motivated by the fact that the energy consumption of a vehicle increases exponentially with velocity, but it is a limited resource. A mobile robot needs to accomplish its mission before the defined deadline by using the limited energy resources carried onboard. It is important to rely on a precise energy model when planning the mission in order to increase the chances of success. This said, an energy model needs to be derived, based on real measurements acquired during an operation using our vehicle.

Collision avoidance is a crucial part of an AUV's ability to perform planned missions in unknown environments. A important feature of an autonomous vehicle is its ability to adjust to previously unknown obstacles along its path, which can pose a threat in the form of a potential collision. The seafloor itself may pose a potential collision threat, especially if navigating in rugged terrain, but so can an harbour wall or even ships manoeuvring at the ocean surface. The loss of a vehicle due to collision is unjustifiable both in terms of cost and replacement time. To prevent this, a robust and effective Collision Avoidance System (CAS) is required. Therefore, it is important to create a system for 3D collision avoidance manoeuvring for the MARES AUV, able to handle obstacles and rough seafloor topography. This will grant the vehicle with a greater degree of autonomy and will also allow to test the robustness and efficiency of the planning methodolo-

gies. Since the MARES AUV does not have a FLS, the focus will be on the development of a CAS for simulation purposes. This includes an AUV model and the motor controllers which were already implemented, a simulated bottom profile, an external navigation system (the INS discussed previously) and sensor simulators.

4.2 Simulator

In order to test all developed subsystems, the MARES AUV simulator (originally created by our colleague Bruno Ferreira), was further developed. The complete model with six degrees of freedom, deduced in (B. Ferreira and Cruz, 2009), given by an nonlinear differential equation, is implemented in C++. This simulation considers forces acting on the body as input at each step and computes the next state. Several restrictions and limitations are considered such as high uncertainty and low frequency due to the horizontal acoustic positioning. One of the main goals of our simulator is that it should be modular and extensible. We do not intend to compare our simulator to others as its development was justified by specific system's engineering requirements. The MARES AUV and its subsystems are developed by our research group. In order to guarantee that these systems operate as expected they need to be tested in an environment as close as possible to the real one. Hence, for these reason the best solution was to create our own simulator, using same communication protocols, and controlling every aspect of the simulation process.

The following features were added in this work:

- noise added to each model output;
- GPS data is sent when vehicle is at surface;
- allow the use of an external navigation process during simulation;
- sends simulated IMU data to external navigation process;
- reads a terrain model from file;
- sends altimeter data, considering the real position of the AUV in the real terrain (simulated position may be different if external navigation process is used);
- added FLS and SSS simulators;
- added a CAS;
- added specification of the individual sampling frequency of each sensor data.

All simulations in this thesis have been performed using this simulator.

4.2.1 System

The simulator block diagram is depicted in figure 4.4. Note that although there are two navigation processes in the system, only one is used during each simulation. This allows the use of navigation data extracted directly from the AUV model or the calculation of data using simulated GPS and IMU data. All interprocess communication is done using UDP message passing. Sensor simulation can be individually disabled since, for example, sonar simulation consumes considerable computing resources and may not always be required.

4.2.2 Sonar simulation

A sonar involves the transmission of acoustic signals or beams. An acoustic beam may be thought of as a continuous wave, spreading out like the ripples after a stone has been dropped into water, or as a number of discrete rays or lines. The reception by the sonar transducer of the echoes produced when these beams hit a target, allows its detection and range estimation. The distance to a target is estimated by measuring the time of flight of a beam and multiplying it by the speed of sound through water. The target's direction is estimated by comparing the time of arrival of each echo received by the transducer.

This subsection describes a computer program which simulates sonar beams by propagating them and processing detections. The system applies the principles of ray tracing from computer graphics to the problems of underwater acoustics. Ray tracing algorithms build a representation of an object in a 3D environment by simulating the path taken by several rays, projected from a light source, as they are absorbed, reflected, or refracted by elements in the scene. The main computational bottleneck is proved to be the discovery of all the intersections of the rays with the environment model, represented by a Digital Elevation Model (DEM) matrix. This calculations are performed at each time-step during simulations and must take into consideration the vehicle's current position and attitude.

The MARES AUV may be equipped with the following acoustic transducers:

- Vertical altimeter: single sonar beam pointing down;
- SSS: a wide sonar beam pointing starboard and a wide sonar beam pointing port;
- FLS: array of sonar beams pointing forward.

4.2.2.1 Sonar beam

As stated previously, sonar beams are simulated by discretizing the beam into several "rays" according to the beam width. Intersections between the environment model cells and the sonar rays are found by comparing the elevation of the iteration points along the rays with the cell's elevation value. Sonar rays are propagated in each simulated step with a constant distance Δd until intersection is detected (iteration point elevation smaller than DEM cell elevation) or maximum range is achieved. In order to facilitate the processing of each sonar line, the propagation calculation is

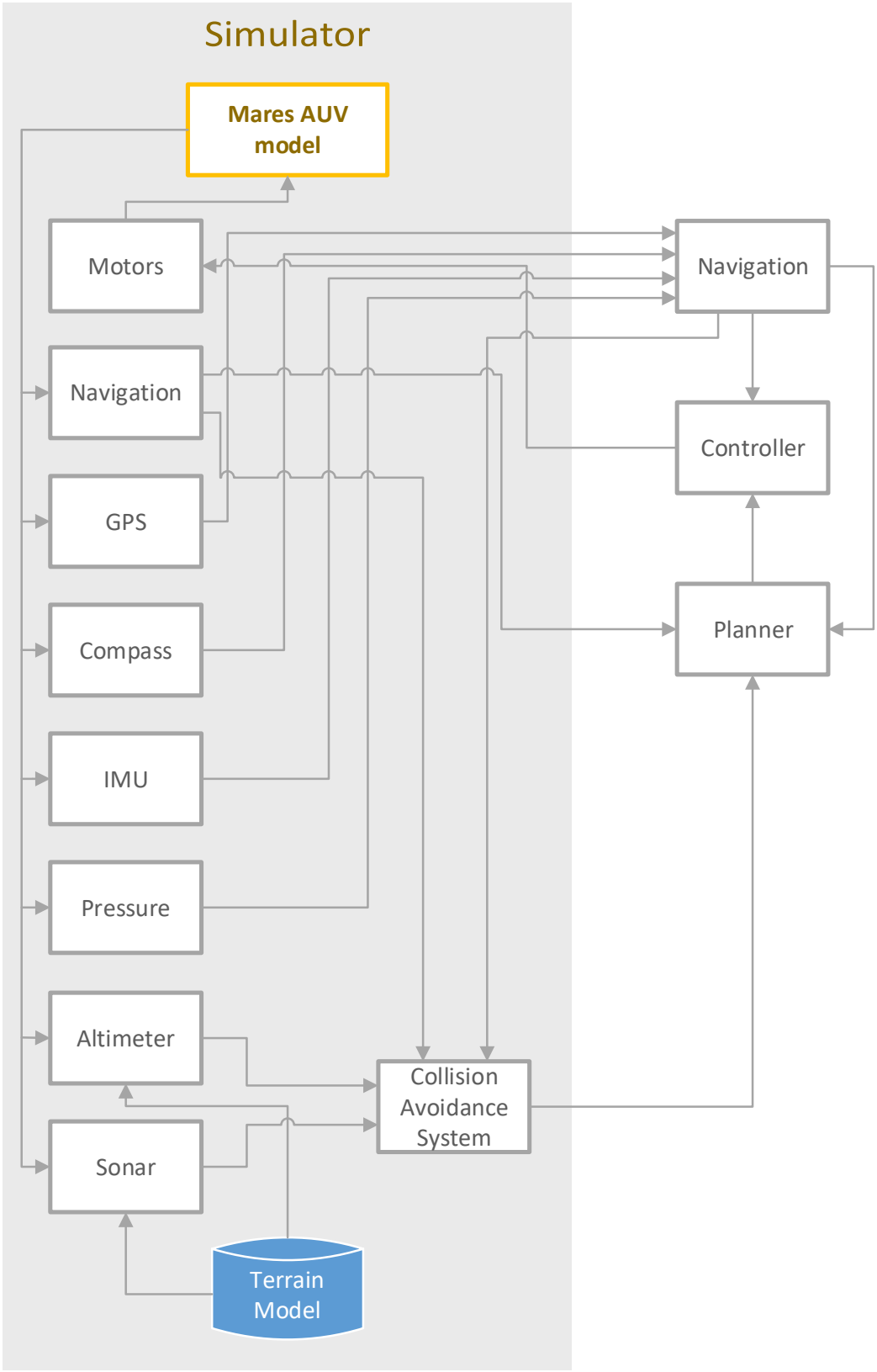


Figure 4.4: Block diagram of the MARES AUV simulator.

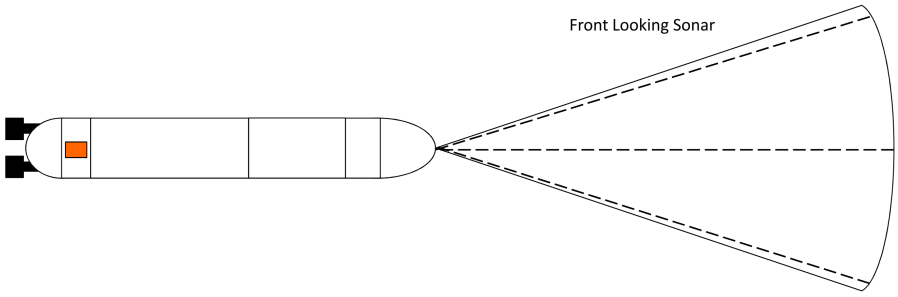


Figure 4.5: Acoustic transducer's configuration (top view).

executed in the sensor frame, then converted to body frame according to the transducer mounting position specified in table 4.1. Finally the position is converted to navigation frame to test if the intersection exists.

The iteration point along the ray in the body frame, p_i^b, is found by applying equation 4.1 where p_i^s is the iteration point in sensor frame and q_l^b is the attitude quaternion representing the ray in body frame. The norm of the iteration point vector is d in any frame where it is represented and this is incremented in each simulation step by Δd. This quaternion is calculated in equation 4.2 by multiplying the attitude quaternions q_l^s and q_s^b, representing the ray's attitude in sensor frame and the sensor's attitude in body frame.

$$p_i^b = q_l^b \times \begin{pmatrix} 0 \\ p_i^s \end{pmatrix} \times (q_l^b)^{-1} \tag{4.1}$$

$$q_l^b = q_s^b \times q_l^s \tag{4.2}$$

Using equation 4.3, the iteration points p_i^b in body frame need to be transformed to the navigation frame, p_i^n, so that the elevation of the iteration point can be compared with the DEM matrix cell's elevation:

$$p_i^n = p_v^n + R_b^n \times p_i^b \tag{4.3}$$

Transducer	**Orientation (roll/pitch/yaw, ○)**	**range (m)**	**nº beams**	**nº lines /beam**	**horizontal width (○)**	**vertical width (○)**
Altimeter	0/-90/0	30	1	1	0	0
SSS (starboard)	0/-45/90	60	1	50	0	60
SSS (port)	0/-45/-90	60	1	50	0	60
FLS	0/0/0	60	45	3	15	45

Table 4.1: Acoustic transducer's configuration parameters.

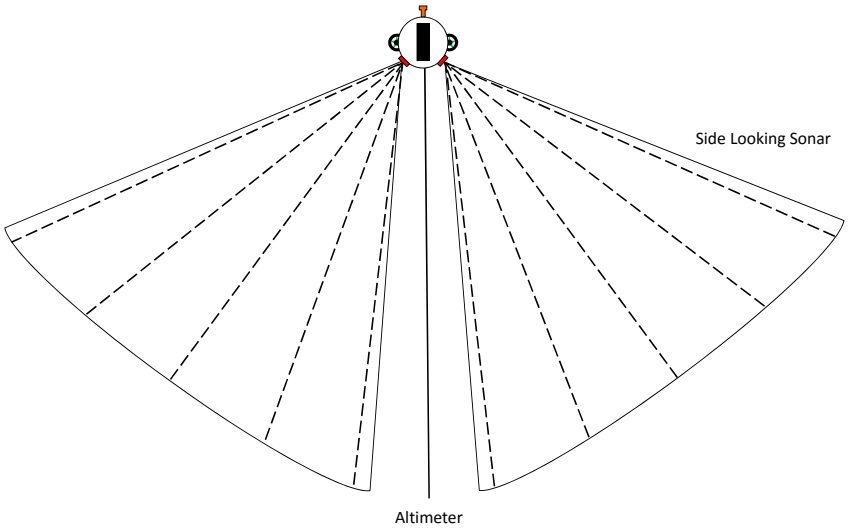


Figure 4.6: Acoustic transducer's configuration (front view).

where p_v^n is the current vehicle's position and R_b^n is the rotation matrix from body to the navigation frame.

4.2.2.2 Sidescan sonars

The two SSS, port and starboard, are identical and therefore are simulated using the technique detailed previously (with the same settings), although the mounting position is different. Each of the sonars emits one wide sector beam with vertical width of 60 degrees, as can be seen in figure 4.6. In order to simulate the beam, it is discretized in 50 rays and each one is tested for any detections.

4.2.2.3 Forward looking sonar

The chosen FLS is pointing forward and emits 45 sonar beams separated vertically by 1 degree, as can be observed in figure 4.7. Each sonar beam has a horizontal width of 15 degrees and is discretized in 3 rays. Each ray is processed as explained previously.

4.2.2.4 Notes

As the sonar simulator performs a great number of calculations in each time step, the simulation speed was significantly reduced after implementing this functionality. Sonar simulators use models that consider important aspects of the underlying physical processes. Each component ultimately needs to tackle the classic speed versus memory trade-off between an highly realistic model, the computational demands imposed by these models and the desired performance. Typically the

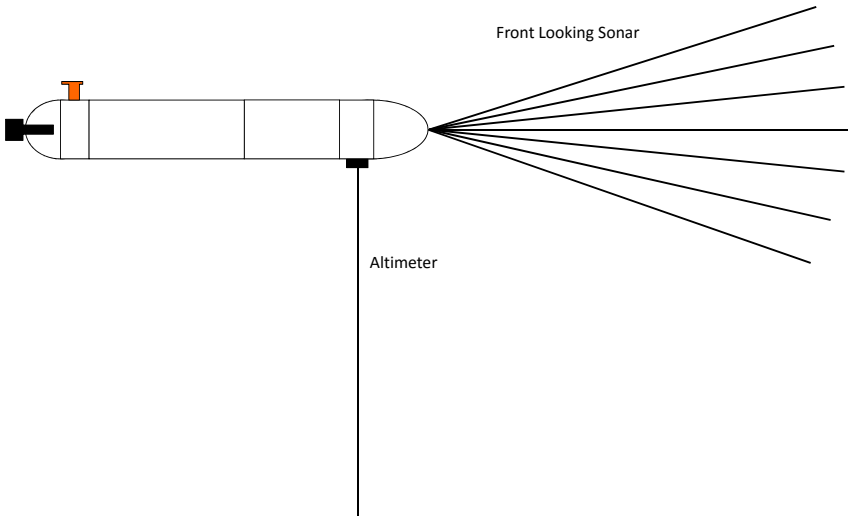


Figure 4.7: Acoustic transducer's configuration (side view).

most computationally intensive component of an acoustic simulation model is the propagation algorithm applied to generate the full temporal and spatial evolution of the acoustic beams and the determination of the intersections with the simulated environment. One of the reasons for this is the large amount of data that needs to be processed when dealing with high resolution DEMs. Realtime application of existing sonar models is restricted by the highly detailed large-scale matrices required to model the seafloor at the level of realism required to achieving accurate acoustic reverberation modeling.

4.3 Inertial navigation system

This section describes in detail the design of an inertial navigation system (INS). Introductions to inertial sensors, coordinate systems and frames of reference are presented.

4.3.1 Sensors

Inertial motion sensing typically uses two types of sensors: accelerometers and gyro sensors. Accelerometers are inertial sensor devices designed to measure linear acceleration. This is executed by measuring the force applied on the sensor and calculating the corresponding acceleration. By calculating the acceleration one can later estimate the relative motion of the vehicle:

- integrating the acceleration gives us the velocity;
- integrating the velocity gives us the position.

Figure 4.8: Inertial Measurement Unit from Xsens.

Gyro sensors are inertial sensors that measure angular velocity. This can then be used to determine the orientation of the sensor by integrating the measured angular velocity.

The MARES AUV is equipped with an IMU, model MTi from Xsens Technologies. Its size is 58 x 58 x 22 mm and it weights 50 grams. A picture of this IMU is presented in figure 4.8. It is configured to provide inertial data at 25 Hz. This IMU has three orthogonally-oriented accelerometers, three gyroscopes and three magnetometers. The accelerometers and gyroscopes are solid state MEMS with capacitative readout, providing linear acceleration and rate of turn, respectively. Magnetometers use a thin-film magnetoresistive principle to measure the earth magnetic field. The Xsens also provides orientation angles with a dynamic accuracy of 2 degrees.

The GPS receiver used here is a Amaryllo Roots receiver that has a sample frequency of 1 Hz. The receiver communicates with the embedded system over USB and utilizes the industry standard NMEA protocol. As for acquisition time, in a cold start situation it takes around 35 seconds to get a fix while in a hot start situation it usually takes 15 seconds. A picture of this GPS receiver is presented in figure 4.9.

4.3.2 Sensor errors

One of the main problems in using inertial sensors for navigation purposes is ensuring the system's reliability as time goes by as a result of error accumulation. To do this it is necessary to identify and understand the common underlying sources of error associated with the inertial sensors used in the IMU.

Figure 4.9: Amaryllo Roots GPS receiver.

- Bias and Drift
  These are the most devastating types of error. A bias is a systematic difference to the true value on the output of a sensor, independent of the input. It won't change during a run, but may vary between "power ups". It is modeled as a random constant. Because bias errors are only constant for short periods, the drift must be considered.

- Scale Factor
  Typically the data acquired from an inertial sensor is proportional to but not equal to the true output. The measured data should be scaled by a scaling factor to obtain the true output. It should be determined experimentally.

- Random Walk
  The output of the sensors is perturbed by white noise which fluctuates at a rate much greater than the sampling rate of the sensor. The Random walk results from the integration of white noise.

- Misalignment
  The non-orthogonality error is the error resulting from the imperfection of mounting the inertial sensors along the three orthogonal axes during manufacturing (mechanical misalignment).

- Temperature
  The IMU's accelerometers and gyros are sensitive to temperature. As the temperature of the IMU changes, the associated bias and drift will change until the temperature reaches steady state or remains the same.

One very important factor is to remove the effect of gravity to obtain the actual acceleration of the object. This can be compensated during the calibration of the system by measuring the actual gravitational force at a specific location and correcting it.

For this application, since the system is only being used for relatively short periods of time, only constant bias is being addressed (it is the most destructive error). Ultimately, the degree of accuracy with which these errors are modeled is dependent on the application as well as the quality of the sensor. It is not possible to compensate all effects. Accelerometer and gyro errors can't be compensated entirely and there are some errors that are not compensated (such as thermal effects).

Before performing a frame of reference transformation on the measured data, it is necessary to remove the constant bias. To accomplish this the output of the IMU is measured over a period of time during the calibration process (at the initialization of the inertial navigation system) to determine the initial state of the device without any external forces being applied. First, the effects of gravity are subtracted from the measurements of the IMU represented in the navigation frame. The constant bias can then be estimated as an average of the data obtained when the device is not affected by external forces and subtracted from the data.

4.3.3 Reference frames and rotations

When doing navigation one can use different reference frames (Barshan and Durrant-Whyte, 1995):

- The inertial frame (i-frame) is a reference frame in which Newton's laws of motion apply. Its fixed in space with its origin located at the center of the earth. All the inertial sensors make measurements relative to an inertial frame.

- The earth frame (e-frame) has its origin fixed to the center of the earth. Its similar to the inertial frame but the axes rotate with the earth. There are two different coordinate systems, rectangular and geodetic, to describe the location of a point in this frame.

- The navigation frame (n-frame) is attached to a fixed point on the surface of the earth at some convenient point for local measurements.

- The body frame (b-frame) is the frame associated with vehicle. Typically the x-axis points to the front of the vehicle, the y-axis points to the right side and the z-axis points downward. The measurements will be taken in this coordinate system.

Each frame has its own advantages and disadvantages so their use is application specific. In this work a fixed navigation frame (n-frame) is chosen for the inertial navigation system analysis so any movement related to the body frame (b-frame) will be converted to n-frame. To transition between the various reference frames, several rotation matrices are needed but, since here the measurements will only be converted from the b-frame to the n-frame, only one is presented. The rotation combination used for the Euler angles here is equivalent to ZYX. C_b^n is the direction cosine matrix and E_b^n is the matrix that transforms the rotation rates in the body frame to the Euler angle rates:

$$C_b^n = \begin{bmatrix} C_\theta C_\psi & -C_\phi S_\psi + S_\phi S_\theta C_\psi & S_\phi S_\psi + C_\phi S_\theta C_\psi \\ C_\theta S_\psi & C_\phi C_\psi + S_\phi S_\theta C_\psi & -S_\phi C_\psi + C_\phi S_\theta S_\psi \\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta \end{bmatrix} \tag{4.4}$$

$$E_b^n = \begin{bmatrix} 1 & S_\phi S_\theta / C_\theta & C_\phi S_\theta / C_\theta \\ 0 & C_\phi & -S_\phi \\ 0 & S_\phi / C_\theta & C_\phi / C_\theta \end{bmatrix} \tag{4.5}$$

where s represents *sin*, and c represents *cos*.

4.3.4 Dynamic model

A state vector X is defined as follows:

$$X = \begin{bmatrix} p \\ v \\ o \\ b_a \\ b_w \end{bmatrix} \tag{4.6}$$

where p = [p$_{north}$ p$_{east}$ p$_{down}$] represents the position, v = [v$_{north}$ v$_{east}$ v$_{down}$] the velocity and o = [ψ θ ϕ] the orientation of the vehicle in the navigation frame. b_a is the IMU acceleration bias and b_w is the IMU gyro rate bias. The dynamic model for the vehicle navigation can be described by:

$$\begin{bmatrix} \dot{p} \\ \dot{v} \\ \dot{o} \\ \dot{b_a} \\ \dot{b_w} \end{bmatrix} = \begin{bmatrix} v \\ C_b^n a^b - (C_b^n w^b + \Omega^n) \times v^n + g^n \\ E_b^n (w^b - \Omega^n) \\ W_{b_a} \\ W_{b_w} \end{bmatrix} \tag{4.7}$$

where Ω^n is the rotation rate of the earth and g^n is the gravitation. a^b and w^b are defined as:

$$a^b = a_i^b - (b_a + W_{b_a}) \tag{4.8}$$

$$w^b = w_i^b - (b_w + W_{b_w}) \tag{4.9}$$

where a_i^b and w_i^b are the acceleration and gyro rate measurements from the IMU. W_{b_a} and W_{b_w} are white noise of acceleration and gyro rate, respectively. In our particular vehicle operation scenario, the vehicle's movement is in a limited geographic area. Therefore, it is believed that the fixed n-frame is sufficient. Since our frame of reference isn't rotating at angular velocity ω, a mass m moving with velocity v won't experience the Coriolis effect and this term will be disregarded in our model. Also, given that our low-cost IMU doesn't have the ability to measure the angular velocity of the Earth, it will not be considered.

4.3.5 Sensor fusion

The discrete Kalman filter (Durrant-Whyte, 2001) is a recursive discrete data filter used to perform state estimation for dynamic systems. It is composed by a set of mathematical equations that are based on the dynamic model of the system. These equations are used to estimate the current state of the system and eventually correct this estimation using any available sensor measurements. Kalman filters have been used in a wide range of fields, particularly in the area of navigation and control systems. These filters are popular because they are:

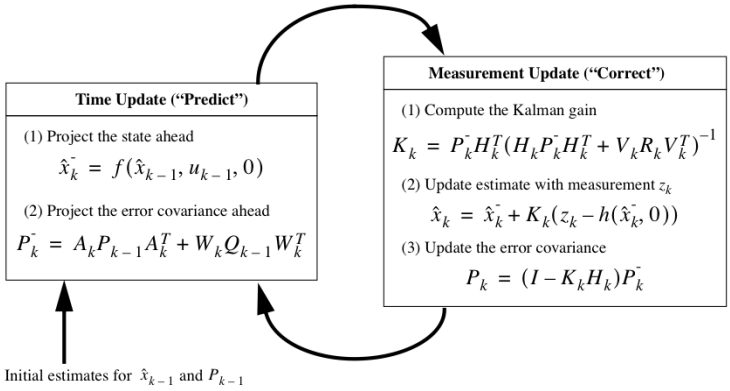


Figure 4.10: Diagram describing the steps involved in the Extended Kalman filter.

- computationally efficient;
- capable of providing good state estimates for the past, present and future;
- able to use dynamic models and error models of the underlying system reducing the effects of noise on the state estimates.

The Kalman filter is addressed in the general estimation problem of state x of a discrete-time controlled process that is governed by a linear stochastic difference equation. If the process to be estimated is non-linear then the extended Kalman filter (EKF) (Welch and Bishop, 2001) is used instead and the process is governed by the non-linear stochastic difference equation:

$$x_k = f(x_{k-1}, u_{k-1}, w_{k-1}) \tag{4.10}$$

with a measurement z that is described by:

$$z_k = h(x_k, v_k) \tag{4.11}$$

where the random variable w_k and v_k represent the process and measurement noise, respectively. They are assumed to be independent of each other, white, and with normal probability distributions $p(w)\ N(0,Q)$ and $p(v)\ N(0,R)$.

In practice, the process noise covariance and the measurement noise covariance matrices might change in each time step. It is assumed here that they are all constant except the GPS measurement noise matrix which is updated depending on the accuracy of the estimated position. For defining the other matrices, the expected variance for the measurements that is registered in the sensors datasheets was used (therefore considered constant during execution of the estimation process).

As the basic Kalman filter, the EKF is also comprises two steps: prediction and estimation. Figure 4.10 shows a diagram that explains its functionality.

In our implementation of the EKF, the following set of measurements is used to update the state estimate:

- GPS position measurements are used to update the north and east positions (in the navigation frame);
- pressure measurements are used to calculate the depth of the vehicle after conversion;
- GPS or Doppler velocity log (DVL) velocity or measurements are used to update the horizontal velocity;
- compass measurements are used to update the current orientation of the vehicle.

4.3.6 Smoothing

When forward filtering is used for state estimation, an optimal smoothing method such as Rauch-Tung-Striebel backward smoother (RTSS), can be applied. The RTSS was first presented by Rauch et al. (Rauch et al., 1965). It was proved as the optimal smoothing method for linear systems on basis of maximum likelihood (ML) criterion. The RTSS has been widely applied in navigation applications due to its robustness and effectiveness. It can be regarded as an (backwards) add-on correction to the Kalman Filter (Gelb, 1974). It uses the EKF estimations as a first approximation. This approximation is improved by using additional data that was not available in the filtering process.

The RTSS is implemented in this study. Compared to other smoothers, RTSS algorithm is the easiest and simplest to implement (Gelb, 1974). The smoothed system state estimate is obtained by using the following set of equations:

$$m_{k+1}^{-} = f(m_k, k) \tag{4.12}$$

$$P_k^s = P_k^+ + C_k[P_{k+1}^s - P_{k+1}^-](C_k)^T \tag{4.13}$$

$$C_k = P_k^+(F_x(m_k,k))^T[P_{k+1}^-]^-1 \tag{4.14}$$

$$m_k^s = m_k^+ + C_k[m_{k+1}^s - m_{k+1}^-] \tag{4.15}$$

$$P_k^s = P_k + C_k[P_{k+1}^s - P_{k+1}^-](C_k)^T \tag{4.16}$$

where f is the model function, m_k^s and P_k^s are the smoother estimates for the state mean and state covariance on time step k, m_k^+ and P_k^+ are the updated filter estimates for the state mean and state covariance on time step k, m_{k+1}^- and P_{k+1}^- are the predicted state mean and state covariance on time step $k+1$, C_k is the smoother gain on time step k, which measures how much the smoothed estimates should be corrected on that time step.

The difference between Kalman filter and Kalman smoother is that the recursion in the filter moves forward and in smoother backward, as can be seen from the equations above. In the smoother the recursion starts from the last time step T with $m_T^s = m_T$ and $P_T^s = P_T$.

The calculation of the RTSS estimates does not involve the smoother covariance. This method allows the smoothing gain to be computed during the forward filter process. Hence, the forward filter covariance as well as the state matrix does not need to be stored.

The computation of the smoothed estimate at each time epoch requires the storage of the EKF predicted and updated (filtered) estimates and their corresponding covariance at each epoch. EKF update steps only take place when measurements are available. When GPS outages occur, like when the vehicle is underwater, only predicted estimates and covariances are available. In any case, IMU data rate is always higher than that of the GPS. These conditions are called measurement gaps. Under these conditions, the EKF only works in prediction mode. The error state and the covariance updates cannot be generated and the smoothing computation is interrupted. One option to solve this problem is to interpret the EKF prediction solution as the temporary update replacement. It is accepted for INS-based integration estimation applications since these predictions are the best approximation when no measurements are available. This replacement is described as:

$$P_k^+ = P_k^- \tag{4.17}$$

$$m_k^+ = m_k^- \tag{4.18}$$

where the subscript k denotes time intervals corresponding to the measurement gaps.

4.3.7 Performance assessment

In order to test the navigation system, a mission containing 4 waypoints was created, forming a box with lateral size of 200 meters. The waypoints were located at different depths and the target vehicle speed was kept constant at 1.5 m/s. Note that, in the experiment presented below, there was no SLAM or map-based localization module running aboard the vehicle. As a result, the pose uncertainty along the vehicle's trajectory can grow without bound. Figure 4.11 shows the location of the vehicle during the mission. The growing error in position from the forward estimator (EKF) is quite visible. Then, when the vehicle returns to surface to get a GPS fix, the smoother (EKS) is correcting past estimates as expected. Figure 4.12 shows a depth profile of the trajectory followed by the AUV. It can also be observed that, when using a navigation system without velocity updates, the vehicle needs to return to surface roughly every 350 seconds. The average time at surface is around 60 seconds, needed to acquire 10 GPS fixes. The request for a GPS fix is executed if the estimated horizontal position variance calculated by the EKF goes beyond a given value.

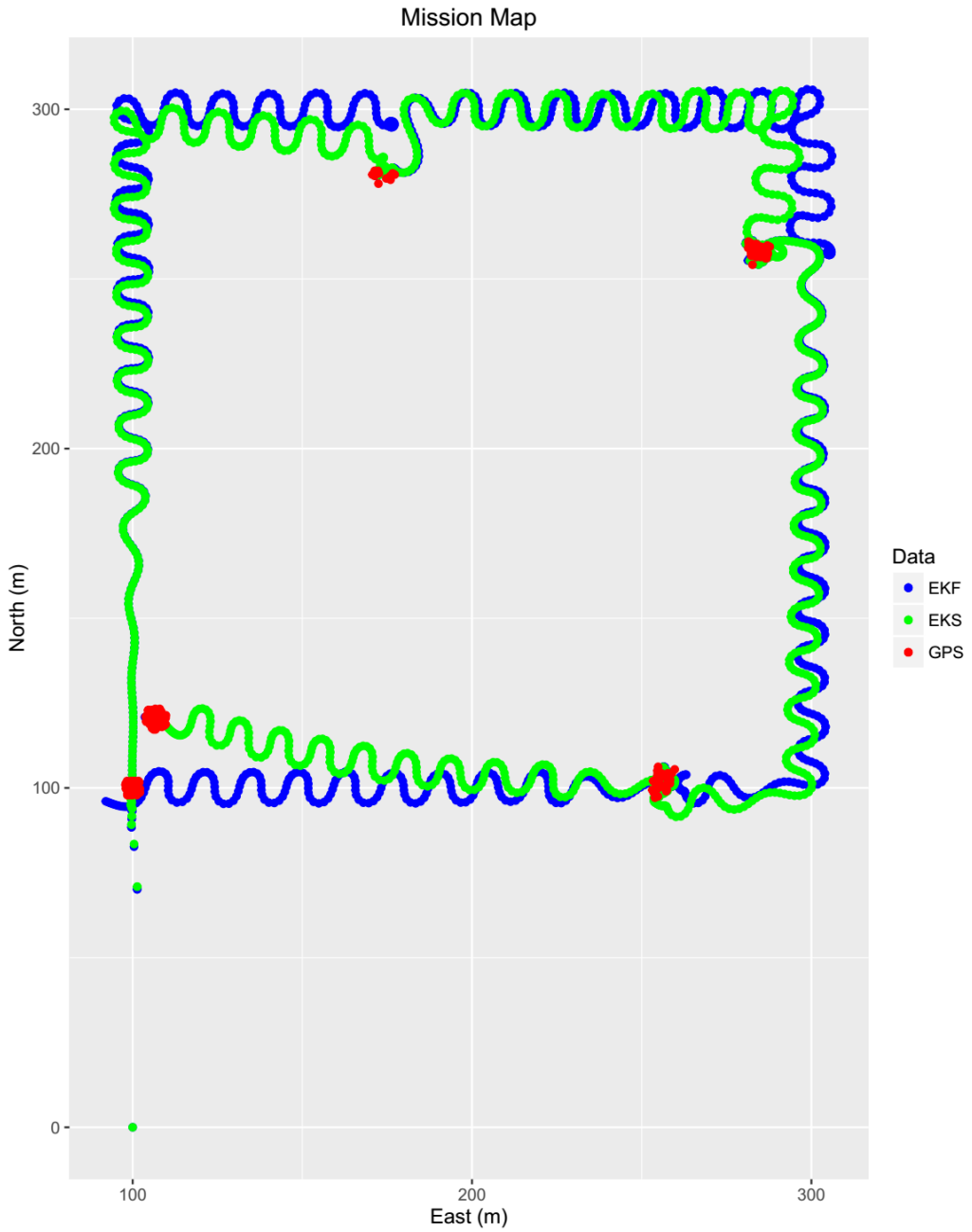


Figure 4.11: Field trial using an AUV and the developed navigation system.

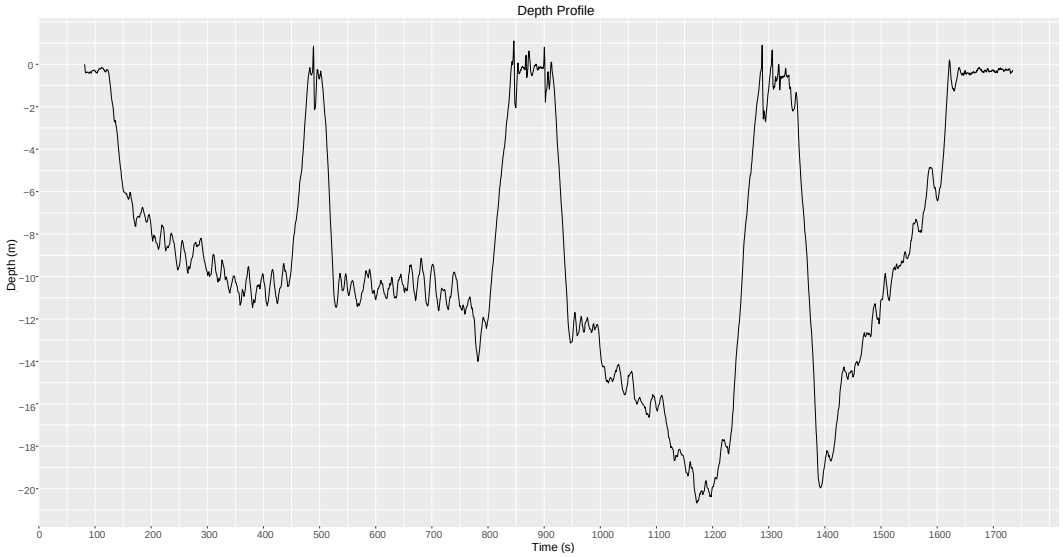


Figure 4.12: Depth profile of the trajectory during the trial.

4.4 Collision avoidance system

4.4.1 System

In general, AUV CASs are usually designed to work together with the existing onboard navigation and planning systems. The CAS is activated only when a future collision is certain if no avoidance manoeuvre is executed. When the initial knowledge on the environment is imperfect or non-existent, sensor based map update needs to be considered at any instant of time. Considering both obstacle mapping and waypoint navigation together is a challenging problem that needs to be addressed before granting the AUV with relevant online planning capabilities to react to changes in the environment. The loss of an AUV due to collision is unjustifiable both in terms of cost and replacement time. To prevent these events, a robust and effective CAS is needed. The goal of the obstacle avoidance algorithms is to avoid collisions with detected obstacles using a local map built from data acquired by acoustic transducers. The simplest acoustic systems use single-beam sonars to detect obstacles in a vehicle's path. More sophisticated systems use multibeam sonars to detect and track obstacles. The CAS should adopt a reactive architecture, handling possible collision situations in an optimal way according to some predefined criteria. A system with optimal behaviour is not likely to exist because of the problem's unpredictability and the high number of potential collision situations. In addition, sensor data is often limited and corrupted with noise and other errors, making it difficult to guarantee a collision free solution. Hence, a primary goal for the CAS is to try to stay on the desired path during collision avoidance manoeuvres and simultaneously try to minimize the deviation in altitude from the previously planned altitude. In the underwater environment, the CAS algorithm should be able to follow obstacles or terrain features when detected and then resume mission afterwards. To this effect, each sensor should be used individually to sense the environment and trigger specific behaviours based on a specified set of rules. After a possible future collision is identified by the sonars, the acquired data has to be

interpreted and combined with the vehicle's navigation data such as position and attitude. Then, a local path planning technique is applied to find a new collision free path based on the original trajectory, the obstacle, the vehicle state and the avoidance strategy.

The intention here is to create a system for 3D collision avoidance manoeuvring for the MARES AUV, able to handle vertical obstacles from ocean floor to surface (e.g.underwater structures) and rough sea floor topography. A CAS is developed and its performance is analysed by integrating the system with the implemented simulator that was detailed previously. This system is based on straightforward principles and known collision avoidance strategies, in order to provide effective, robust and predictable performance. The proposed system provides feasible solutions during simulation and the collision avoidance manoeuvres are executed in accordance with the specified requirements. The developed simulator and collision avoidance system is expected to provide a suitable framework for future work and possibly a implementation on the MARES AUV. The developed CAS should be able to:

- avoid collision and the infringement of a minimum distance to any object;
- minimize horizontal cross track error from the planned waypoints;
- minimize errors from the desired altitude on the current leg.

As a consequence of the overall system goals, the collision avoidance strategy was derived:

- attempt to "fly" over any detected obstacles up to a specified safety distance above the sea floor;
- if this is not possible, go around the obstacle keeping a minimum distance to its boundary (edge following).

The Bug algorithms (Lumelsky and Skewis, 1990, Lumelsky and Stepanov, 1987) are important path planning approaches currently used for AUV collision avoidance purposes, more specifically for edge following. They follow a simple common sense approach of moving the vehicle directly towards the goal waypoint, unless an obstacle is found, in which case the obstacle is contoured until motion to goal is again possible. Their simplicity is a major advantage of these algorithms as they don't need any a priori information about the environment. In addition, they will find a solution if it exists.

The "Bug algorithm" is the simplest collision avoidance algorithm is called "the Bug algorithm" (Lumelsky and Skewis, 1990). According to it, when an obstacle is encountered, the robot fully circles the object in order to find the point with the shortest distance to the goal, then leaves the boundary of the obstacle from this point.

The updated version, known as "Bug 2 algorithm" (Lumelsky and Stepanov, 1987), commands the vehicle to head towards the goal on the mission line. If an obstacle is encountered, the AUV is commanded to follow it at a predefined distance until the vehicle crosses the mission line again. In this case the vehicle moves away from the obstacle and continues towards the waypoint on the mission line. So, the two versions of the algorithm differ on the conditions under which the

border-following behaviour is changed to the line tracking behaviour. In general, Bug 2 algorithm has a shorter travel time than Bug 1 algorithm and is more efficient specially in open spaces.

However, it is believed that the original Bug algorithm is the right choice for our operational scenario. Once an obstacle is detected with the onboard sensors, the complete boundary of the obstacle needs to be identified. Continuing the mission without fully detecting the boundary of the obstacle may imply another boundary following phase in the future which increases the inefficiency of the mission. Instead, after completely defining the obstacle, the mission replanning procedure is triggered so a new mission can be obtained, excluding the new area belonging to the obstacle.

As stated above, if the vehicle fails to "fly" over the obstacle, the *edgefollowing* behaviour will be activated. The *flyover* behaviour is interrupted if steep terrain is detected by the FLS or if the vehicle is requested to "fly" over a predefined minimum depth. The terrain is classified into high-slope and effectively planar regions by analysing the rays from the FLS data. The data is binned according to yaw direction and the slope is calculated for each bin. Then, it is compared with a user-defined slope threshold to obtain an binary classification. If slope is higher then the threshold then the *edgefollowing* behaviour is activated (turning left) and the vehicle tries to detect regions with higher slope using the SSS data. During the initial collision avoidance manoeuvre that applies the *flyover* behaviour, if the vehicle, after analysing FLS and altimeter data, is forced to change to a depth smaller then a given predefined minimum depth then the *edgefollowing* behaviour will be activated. In this case, after the turn is completed, the system will analyse the SSS data and identify detections that belong to the obstacle at the specific terrain depth that was measured when the behaviour was switched. In any case, after the detections are processed a new waypoint is calculated guiding the vehicle around the obstacle. As a consequence of introducing the depth limit, the vessel must circumvent any obstacle present at this depth. The waypoint is calculated by performing orthogonal regression on a set of consecutive detections. This behaviour ends when the vehicle returns to the initial position or when it reaches the boundary of the operational area. Then a replanning task is executed. During edge following the vessel should maintain the desired distance to obstacle not only to the side, but also downwards. This implies that the *flyover* behaviour remains active but instead of trying to track the line that belongs to the predefined path, it just follows the waypoints around the obstacle. The minimum depth parameter was introduced for safety reasons, enabling the vehicle to navigate while trying to minimize the probability of collision with surface vessels in the area. Figure 4.13 shows a diagram that describes our CAS algorithm.

4.4.2 Experiments

The described CAS was evaluated in a simulated but realistic environment to test its functionality and robustness. The simulated terrain is unusually rugged compared to the normal operating areas that are found during a mission with the MARES AUV. This is required to force the activation of all modes of operation and test its behaviour. The terrain has steep gradients transitions which the vehicle is not capable of "flying" over maintaining the desired altitude, due to the kinematic

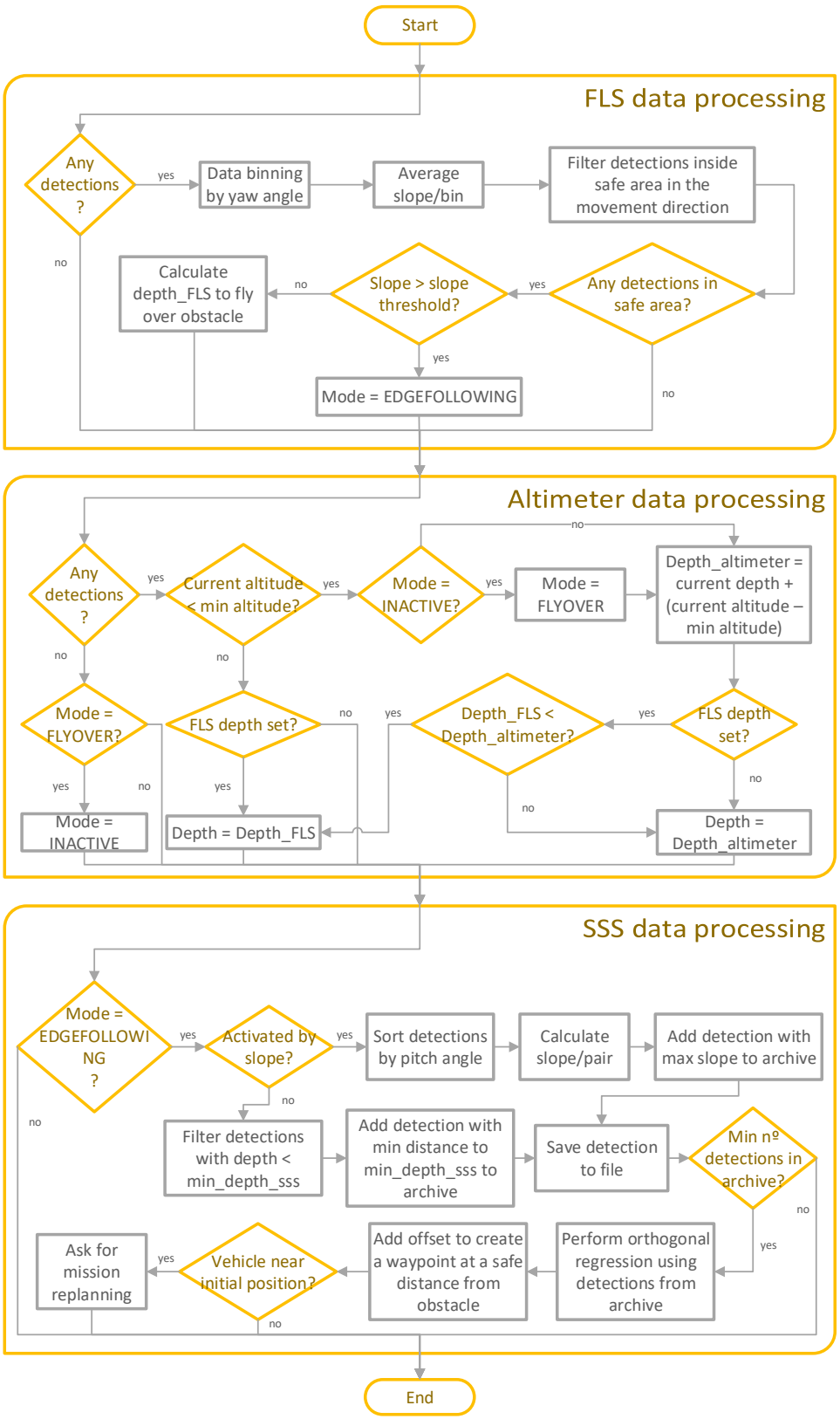


Figure 4.13: Collision avoidance system time step algorithm.

Parameter	Value
Minimum altitude altitude (meters)	5
Minimum depth (meters)	2
Critical distance (meters)	10
flyover slope limit (^{o})	80
SSS history min size (detections)	10
SSS history max size (detections)	20
Time per waypoint (seconds)	30
Distance to termination (meters)	5

Table 4.2: Collision avoidance system configuration parameters.

constraints. A CAS is essential for terrains with similar features in order to satisfy the mission objectives and to successfully avoid collisions. When navigating in less rugged terrains, the AUV does not require full CAS functionality to operate safely, therefore simulating such conditions is not ideal when evaluating the system. The simulations have been performed with the parameters presented in 4.2.

The simulation starts at 200 meters East and 0 meters North and the mission is to perform a rectilinear transect heading north. The *flyover* mode is activated at 160 seconds after detecting steep terrain by the FLS, avoiding the collision by "flying over" the obstacle. The manoeuvre is executed with pitch close to 0 until the desired depth is achieved due to the existence of 2 vertical motors on the MARES AUV. At 240 seconds, the CAS asks the vehicle to reduce depth below the specified minimum depth in order to avoid a collision with the islet and as a consequence the *edgefollowing* mode is activated. The vehicle then turns left and starts following the obstacle boundary (with data from the SSS) at the specific depth of the terrain at time of activation with reasonable performance. The realistic features of the terrain created by the fractal terrain simulator and its ruggedness made the detection of the desired boundary more complex, but as can be seen in the plot from figure 4.15, the vehicle was able to circumvent the detected obstacle. The simulation ended when the vehicle achieved the boundary of the area of operation. The next step would be to define the area occupied by the detected obstacle, perform the decomposition of the operating area excluding that area and to replan the mission for the resulting areas.

4.4.3 Conclusion

In all the simulations the CAS performs well and acts in accordance with the overall goals and user requirements. However, due to the complexity of the collision avoidance problem, it is very difficult to design a system capable of handling every thinkable situation. By the same reason, there are a number of situations in which the developed system will not be able to avoid collision. An example is the AUV getting trapped within a subsurface cave. However, it is assumed that the developed system will be able to handle most situations by adding functionality for special cases. For this reason, the developed system should be considered only as a concept of collision

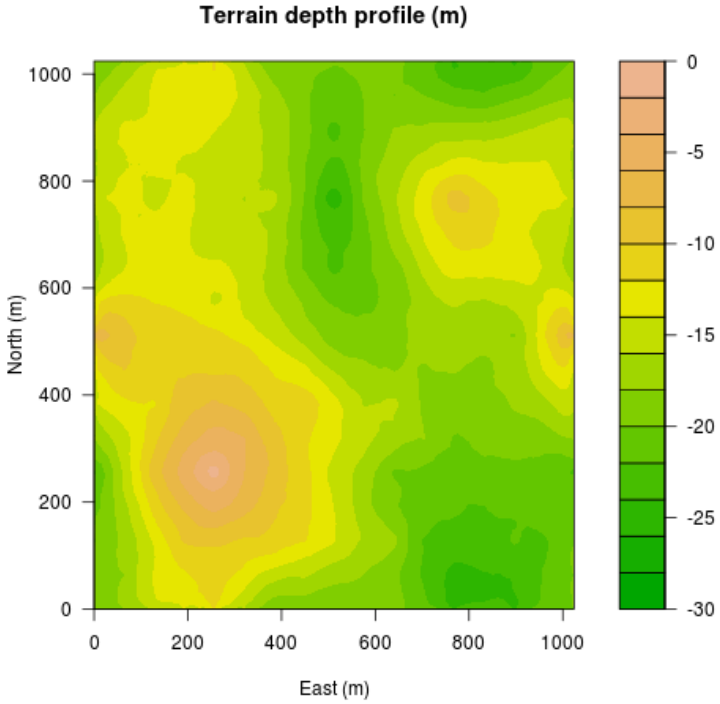


Figure 4.14: Simulated terrain depth profile.

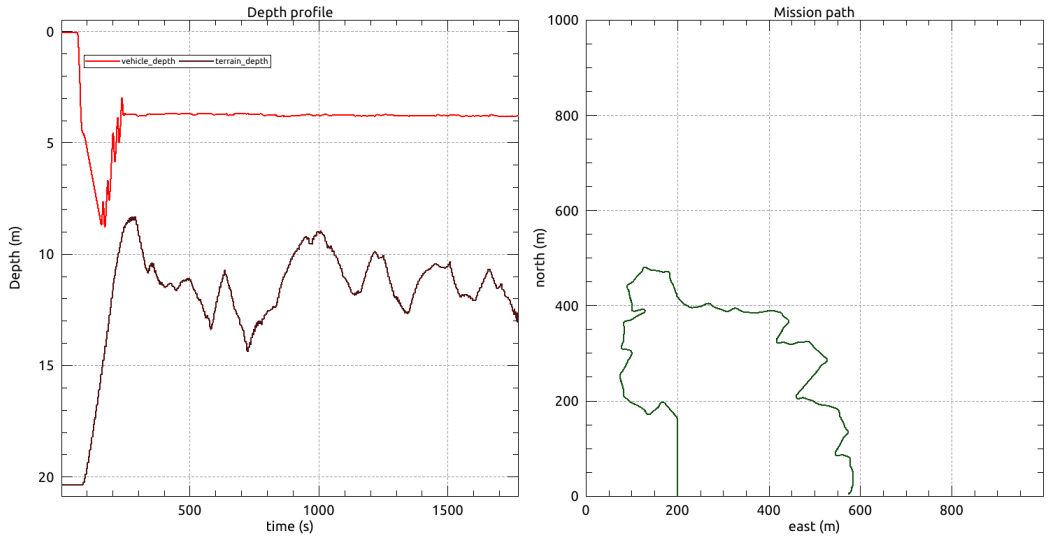


Figure 4.15: Plots displaying the vehicle's depth profile and path taken during the mission, respectively.

avoidance. There is still a number of issues that needs attention before the system is ready for deployment.

4.5 Energy Model

Estimating a vehicle's energy consumption along a predefined trajectory is a challenging task (Sadrpour et al., 2012) considering the variety of vehicles that exist today with different characteristics and dynamics. The high number of degrees of freedom (DOF) and the variability of the operating conditions makes this estimation even more difficult. Several aspects should be considered such as the effect of external disturbances, which usually are not taken into account in the dynamic models, and the performance of the motion controllers that are used in the vehicles. Since this is not the main goal of the thesis, other existent models for energy consumption will not be reviewed here. The chosen approach, detailed next, satisfies our requirements and provides a decent approximation to the real energy consumption of the vehicle during a mission.

A procedure that enables the determination of an energy model for any underwater or surface vehicle needs to be defined, abstracting from the underlying control system while only analysing their motion and energy expenditure under specific operating conditions. It is imperative to maintain the computational complexity of the energy model low enough to allow its use by a online path planner. This said, a linear energy model for estimating the energy requirements for a specific path will be defined. Consider a path P, given as an ordered list of n waypoints described by a vector in NED reference frame. The energy is calculated by analysing sequentially every segment of the trajectory uniting pairs of waypoints:

$$E[P] = \sum_{i=0}^{n-1} \Delta E[i, i+1] \tag{4.19}$$

When analysing each segment, one needs to consider the vehicle's state variables variability when moving from its current state to the desired state:

$$\Delta E[i, i+1] = k_x \Delta x + k_y \Delta y + k_z \Delta z + k_\phi \Delta\phi + k_\theta \Delta\theta + k_\psi \Delta\psi, i = 0, 1, ..., n \tag{4.20}$$

where Δx, Δy and Δz represent the displacement on the longitudinal, lateral and vertical axes, in the body frame, and the $\Delta\phi$, $\Delta\theta$ and $\Delta\psi$ represent the rotations around these axes. Calculating the displacement vector for the state variables can be easily done by subtraction, obtaining the sequential relative change of position and attitude at each path segment. The coefficients k_x, k_y, k_z, k_ϕ, k_θ and k_ψ in equation 4.20 are estimated by performing a set of experiments designed to test one degree of freedom at time. For the specific case of the MARES AUV, the surge movement is caused by the motors assembled horizontally while the heave is caused by the motors assembled horizontally. For simplification purposes, it is assumed that the motors assembled vertically are only used to control pitch between consecutive path segments and, therefore, will ignore any need for correction while travelling along the segments. Although the MARES AUV can rotate around its own axis, this manoeuvre is not considered here since this manoeuvre is not expected to occur

during a coverage mission. Furthermore, since roll can't be controlled, the corresponding terms in the model will also be ignored. During the experiments, the energy usage is estimated by integration of the instantaneous power consumption monitored by the smart battery controller at a sampling rate of 2 Hz.

4.5.1 Experimental procedure

Power vs. Velocity

Here the objective is to derive the power consumption as a function of the vehicle's velocity while performing specific manoeuvres such as motion in the surge and heave directions and hovering. The following procedure was suggested:

1. Surge experiments: test a minimum of 4 distinct velocities; for each velocity execute a straight line trajectory at constant depth;

2. Heave experiments: climbing and descending movement is always performed at a constant velocity; dive to a given target depth and return to surface;

3. Hovering experiments: can be performed together with the previous experiment by hovering at the target depth for 30 seconds.

From these experiments, create a look-up table using the pairs of values that were registered. Then perform a regression to find a function that better represents this relationship, enabling the interpolation for any requested velocity value.

Power during rotations

Here the intention is to measure the time and power needed to perform a rotation. The following procedure was suggested:

- Yaw rotation: test a minimum of 4 distinct velocities; for each velocity execute a trajectory with 3 waypoints at constant depth, defining 2 segments with an angle in the horizontal plane.

By subtracting the registered power consumption here with the one calculated from the previous experiment (while following a trajectory with the same total length) an estimate of the extra energy and time required to perform the rotation is obtained.

4.5.2 Results

By performing the above experiments in real field tests, detailed information on power consumption was acquired. The obtained exponential model for the surge movement can be visualized in

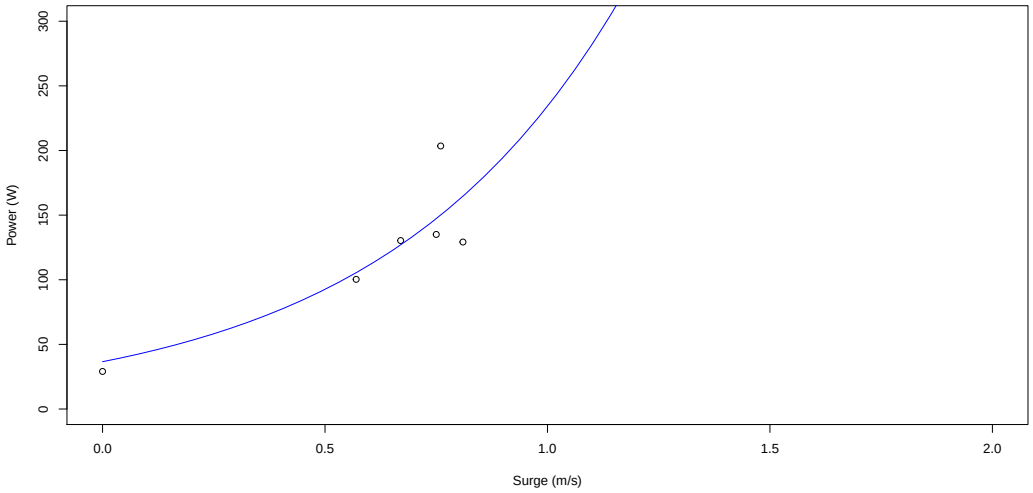


Figure 4.16: Energy model for the surge movement.

figure 4.16. Equation 4.21 represents the derived model. Note that due to the lack of measurements at higher velocities, the determined model is not considered to be accurate for velocities above 1 m/s. Thus, these experiments will continue to be performed (as the navigation system performance is being improved) and the model will be updated accordingly.

$$p_x(v) = e^{-11.496+1.855v} \tag{4.21}$$

Analysing the data from the rotation experiments was more complex. When approximating a corner, the vehicle reduces the velocity (thus reducing energy consumption) to avoid overshooting and then, at the exit, the velocity is increased again. By looking at the power profile it was possible to identify instant power spikes at the exit of the corners, which could be up to 3 times the average power. However, in these experiments it was not possible to assess the effect of velocity while rotating due to the variability of the measurements. Therefore, it was decided to consider an average value on the measured power consumption while performing similar rotations. A constant angular velocity was also assumed while turning. The estimated values for the model parameters are presented in table 4.3.

Since it was not possible to determine the effect of velocity on energy consumption while turning, the validity of the function for k_ψ needs to be evaluated. This parameter is obtained by subtracting energy consumption while turning (90 degree turn) to the energy consumption while moving forward, for the same amount of time. By observing the plot in figure 4.17, it could be seen that as velocity is increased, the energy required for turning decreases, which is false. Then, considering that an average value of energy for turning is being used, a maximum value for the velocity needs to be established. Looking at the plot, it seems a maximum value of 2 m/s is acceptable.

Using this factors, the energy consumption for a given mission path can be easily estimated

Factor	Value	Description
k_x	$\frac{p_x(v)}{3600000v}$ (kWh/m)	energy to move 1 meter forward at velocity v
k_z	$0.20111x10^{-3} \quad z<0$ $0.12611x10^{-3} \quad z>0$ (kWh/m)	energy for diving or surfacing 1 meter
k_ψ	$1.55747x10^{-5} - \frac{p_x(v)}{15509817}$ (kWh/^{o})	energy for turning 1 degree
k_{hover}	$1.73921x10^{-5}$ (kWh/s) (62.6 W)	energy per second to keep the vehicle hovering at a given depth
$k_{payload}$	$0.80750x10^{-5}$ (kWh/s) (29.1 W)	vehicle payload energy requirement per second

Table 4.3: Energy factors derived from the measurements performed during the experiments.

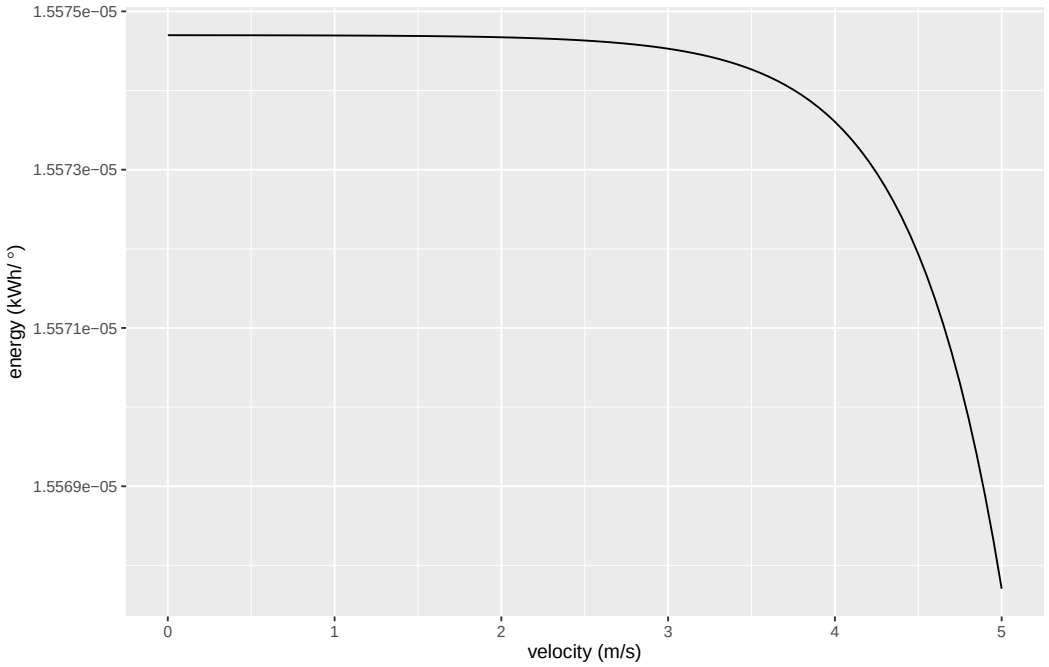


Figure 4.17: Plot for the turning energy factor k_ψ.

by multiplying the factors with the corresponding displacement variable. In the future, we expect to be able to estimate a penalty in time taken by the vehicle when cornering, in addition to the penalty in energy which was determined now.

Because of its simplicity, the proposed model can be adjusted before performing a new mission using a short calibration procedure to adapt the model coefficients to different external payload configurations which can introduce effects, such as increased inertia and drag, on vehicle performance.

Chapter 5

Offline mission planning

Most challenging real-world optimization problems involve multiple objectives. In a multi-objective optimization problem, the goal is to find a set of different solutions representing distinct trade-offs among the different objectives. These solutions are known as Pareto-optimal solutions (Deb, 2001). A solution that is good with respect to one objective may require a compromise in other objective. This limits the ability of choosing a solution which is optimal with respect to only one objective. In problems with more than one conflicting objective there is no single optimum solution. There exists a number of solutions which are equally optimal and without any further information about the problem or about the decision makers preferences, no solution from this set can be said to be better than the other. This set of solutions can be found using the Pareto Optimality Theory (Deb, 2001). The goal of multi-objective optimization is to find a set of solutions close to the Pareto-optimal solutions, diverse enough to represent the entire spread of the Pareto-optimal front. Pareto dominance-based techniques remain the most popular selection scheme adopted by multi-objective evolutionary algorithms (MOEAs), because of the several advantages that it provides. The SPEA 2 (Zitzler et al., 2001) and NSGA 2 (Deb et al., 2002) are two of the most popular MOEAs used when comparing a newly designed MOEA.

Our approach is a a posteriori technique, more specifically a Pareto-based approach, that on each iteration searches for a set of solutions near the true Optimal front. Initially, a *standard* version of the EA was developed by adapting SPEA2 to the specific problem being addressed here. Then, an *informed* version was proposed addressing some of the disadvantages of the previous version, namely the speed of convergence and overall quality of the obtained solutions. Since the algorithms are similar in nature, it was decided to present just the informed version and compare both approaches by performing a series of experiments. Next, the components of the proposed MOEA[1] will be described. Note that this mission planning algorithm is an offline planner that should be executed before the vehicle is deployed in the target area.

[1] the work presented in this chapter was published in (Abreu and Matos, 2014a,b).

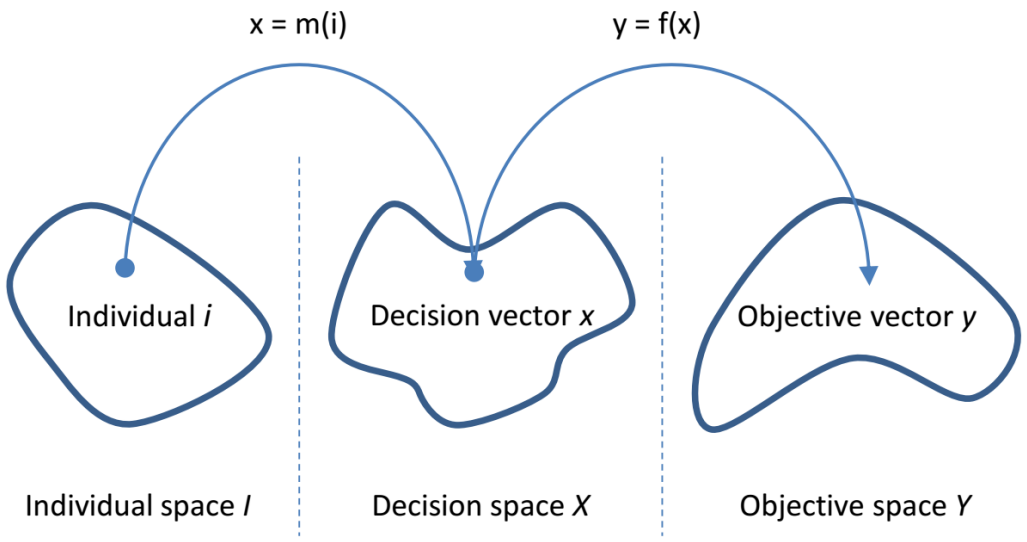


Figure 5.1: Relation between individual space, decision space and objective space.

5.1 Individual representation

The most important aspect when designing a EA is the definition of the representation of an individual. Choosing a good representation that is descriptive and evolvable is a hard problem in evolutionary computation. An individual is considered to be a solution represented by a vector of real valued parameters $x \in R_n$. Thus, an individual is represented by:

$$Ind \in (t, \theta, d, a, v) \tag{5.1}$$

where t is the track spacing, θ is the track direction, d is the depth of the path (when planning with constant depth), a is the altitude of the path relative to the seafloor (when planning with constant altitude) and v is the vehicle velocity. This set of parameters is needed to automate the procedure of generating different individuals in decision space. They also allow the comparison of high level proprieties of distinct individuals. These parameters are related to decision variables and therefore can be used to derive values for these variables.

The quality of an individual with respect to the optimization problem is represented by a scalar value known as $fitness$. Since the quality is determined by the objective functions in the presence of constraints, an individual needs to be "decoded" in order to calculate its fitness. This concept is illustrated in figure 5.1. Given an individual $i \in I$, a mapping function m derives the decision vector x from i. Then, the value of the objective function $f(x)$ is obtained and the fitness value can be assigned to i.

5.2 Algorithm

Realistic and complex engineering problems, like the one being discussing here, involve objective functions that can be very expensive computationally. Consequently, it is advisable to assess the integration of approximate models, which are usually orders of magnitude faster. These approximate models provide less-accurate but more efficient estimates of the original objective function values and can be either readily available or can be learned online (during the course of the optimization) or offline (by sampling and building a model before optimization). This can be interpreted as the addition of a learning mechanism because the information acquired due to the interaction with the environment will allow the acceleration of the evolutionary process.

Our approach combines an EA with an approximation technique, more specifically ANNs. EAs usually need a sufficiently large number of function evaluations to reach the optima of the problem. The approximate model is used to reduce the number of expensive exact function evaluations. Since the model is not known a priori, the initialization or even the initial generations would need to use the exact function evaluations. As the goal in this initial stages is to build an accurate model of the fitness landscape, a large initial population size is required with the individuals distributed throughout the decision space. The main task of the approximate model is to capture the general trend of the fitness landscape and guide the search towards the better regions while reducing the number of exact function evaluations needed to find the best solutions. This guidance is executed by assessing the performance of multiple candidate individuals using the approximate models, with a much smaller evaluation time, and choosing only the best to integrate the next generation population. As generations progress, the population tends to converge towards the better regions of the approximated fitness landscape. One of the challenges that arises is the need to update the approximate model using new information from exact function evaluations throughout the execution of the algorithm, adding finer details to the fitness landscape.

The performance of the neural network is closely related to the efficiency of the training process and the quality of the training data. It is important to develop an algorithm that adjusts the training process to the performance of the neural network in an changing environment. Building a ANN with adequate architecture, training parameters and enough data, makes it possible to achieve a sufficiently small training error. If the data used for training is not representative of the decision space, then significant estimation errors may take place. This can easily happen if the training data is insufficient, which is common during the initialization phase of the EA as the number of exact function evaluations is very limited in order not to take too much time. As the number of generations increases, more exact function evaluations will be available covering different regions of the search space. For this reason, the dynamic process of training a ANN is understood as online learning. The training strategy is difficult to delineate as the minimization of estimation errors is needed while maintaining the process sufficiently fast in order to reap the benefits of using approximate models in optimization algorithms. Another difficulties are how to improve the generalization capability of the network and manage a continuously growing training data set without running out of memory or increasing the processing time beyond given limits (see

subsection 5.7). The number of exact function evaluations required to train the network is closely related to the properties of the functions being approximated. If the functions have discontinuities or large oscillations, the number of samples required to train the network will be higher.

Algorithm 1: Evolutionary mission planner

```
1  function EvolutionaryPlanner
2  Population = Initialize();
3  while !terminated do
4  |   Evaluate();
5  |   TrainNN();
6  |   FitnessAssignement();
7  |   EnvironmentalSelection();
8  |   CheckPerformance();
9  |   generation++;
10 |   terminated = CheckTerminationConditions();
11 |   if terminated then
12 |   |   break;
13 |   end
14 |   MatingSelection();
15 |   RecombinationNN();
16 |   MutationNN();
17 end
```

Initialisation

- Randomly generates N_p individuals from a uniform distribution in decision space;
- Uses them to train the approximate model.

Neural network training

- Uses a classical multilayer feed-forward ANN using the standard error back-propagation algorithm;
- The ANN receives in the input six parameters that define an individual, and generates the estimated objective function values at the output;
- Executed every n_t generations;
- Since a ANN with two hidden layers can approximate any function with negligibly small error (Huang, 2003), the networks are tested with different number of neurons in each layer and the best topology is chosen (smaller mean square error in cross-validation).

Fitness assignment

- Same procedure as in SPEA2.

Environmental selection

- Uses variable archive and population sizes (avoids creating clusters of individuals);
- Controls the individual's density (in objective space) in the containers. Density is controlled by establishing a minimum distance between individuals. This distance is measured in the objective space using the Euclidean distance. That is, the distance between two individuals x and y is calculated as:

$$|i(x) - i(y)| = \sqrt{|i_{POD}(x) - i_{POD}(y)|^2 + |i_E(x) - i_E(y)|^2 + |i_T(x) - i_T(y)|^2} \tag{5.2}$$

  where POD, E and T are normalized values of the objective functions obtained for each individual.

Termination

- The EA stops when a prespecified number of generations is achieved, when the maximum execution time is achieved or when it converges.

Mating selection

- Mating occurs between individuals in the archive and between individuals in the archive and the population, hoping that new individuals are closer to the true Pareto front;
- The pair should be close to each other in objective space to minimize the randomization effect.

Recombination

- Evaluates candidate offspring using the approximate model (cheaper);
- Ranks them in terms of estimated improvement over the less fit parent (choosing one objective function);
- Evaluates the best one using the exact objective functions and adds it to the population;
- Should significantly speed up the EA because it can quickly test multiple candidate individuals.

Mutation

- Evaluates possible mutations using the approximate model (cheaper);
- Performs sensitivity analysis using the ANN, testing different variations of the original individual and analysing the estimated performance;
- Ranks them in terms of estimated improvement over the original individual;

- Evaluates the best one using the exact objective functions and adds it to the population.

5.3 Evolutionary performance evaluation

To properly evaluate and analyse EA performance, three different properties should be studied:

5.3.1 Convergence

The evolution of the search process is assessed by analysing convergence. This is done by comparing sets of individuals from consecutive generations and determining the number of individuals that are dominated in each set (Goel and Stander, 2010). In elitist algorithms, the archive at a generation is compared with an older archive. The number of solutions in the old archive that are dominated by the current archive and the number of older archive members that remain in the current archive should be computed as such measures indicate the improvement in the quality of solutions. The comparison of two different archives should be more straightforward if the number of dominated solutions is scaled by the size of the archive. The smaller the number of dominated solutions, the better is the convergence. This convergence metric is important in cases where a priori knowledge of the Pareto-optimal front is not available, like in the problem we are addressing here, implying that the accuracy cannot be assessed.

5.3.2 Uniformity

The uniformity metric estimates how uniformly the individuals are distributed in the Pareto front, which is a requirement since one of our goals is to acquire a set of solutions that cover the entire Pareto-optimal region. The uniformity metric is defined as (Deb, 2001):

$$\Delta = \sum_{i=1}^{N} \frac{|d_i - \overline{d}|}{N} \tag{5.3}$$

$$\overline{d} = \frac{1}{N} \sum_{i=1}^{N} d_i \tag{5.4}$$

where d_i is the crowding distance of an individual in objective space. Crowding distance is defined as half the perimeter of the largest hypercube around a particular individual without surrounding any other individuals (Deb, 2001). In order to compute Δ, the crowding distance between consecutive non-dominated individuals (obtained by ranking the individuals according to each objective function) needs to be calculated. The average of these distances $\overline{d}$ also needs to be calculated. The boundary points are assigned a crowding distance of twice the distance to the nearest neighbour. It combines the values of different objective functions but, since they are normalized, the units are irrelevant. This measure is similar to the standard deviation of the crowding distance, therefore a small value of the uniformity metric characterizes a good set of solutions.

5.3.3 Volume

This metric measures how wide-spread the individuals in the Pareto front are. It is calculated as the volume of the largest hypercube in objective space that contains all individuals. A large spread is desired to obtain a bigger trade-off surface. As the volume (or spread) metric is determined considering just the individuals located at the boundary of the Pareto front, it is very sensitive to the presence of isolated points that can artificially improve it. Therefore, diversity can only be assessed by performing a combined analysis of the uniformity and volume metrics.

5.4 Local optimization

In this section, motivated by the efficiency of both evolutionary global search and local search strategies, the previous algorithm is extended into a multi-stage multi-objective EA. This hybrid algorithm combines the strengths of both evolutionary and local search, in particular of EAs and SA. To this date (2014) Simulated Annealing (SA) has never been combined with an EA for solving coverage problems. SA gives us the opportunity to quickly improve any solutions that are being considered, working as an unrestricted mutation operator in order to move to new non-dominated solutions. Unrestricted because in this stage the algorithm does not consider the representation of the individual used in the evolutionary process and, therefore, is able to fine tune any solution found by the previous stage without compromising the efficiency of the evolutionary search process.

This local search stage, based on SA, has been designed to optimize two distinct objectives, namely:

- Minimize the uncovered area;
- Maximize the average POD.

These objectives are prioritized differently according to our specific mission planning goals: the average POD can only be increased if there are no more uncovered areas. Note that decreasing the amount of uncovered area will lead to an increase in average POD but not to its maximization. The basic idea behind the algorithm is to try to locally improve the population of non-dominated solutions found by the EA (the solutions in the external set are the initial solutions to this procedure). This is a combinatorial problem where the best set of inter-track distances needs to be chosen, maximizing the coverage of an area characterized by a specific topography. Distinct strategies are used depending on the type of sonar that is being used for seafloor mapping, more specifically if using multibeam (no nadir gap) or sidescan sonar (with nadir gap).

5.4.1 Strategy for the case without nadir gap

Initialization

- Determine the original spacing between tracks (kept constant by the EA);
- Neighbourhood size is set to the initial spacing.

Evaluation

- Determine the amount of uncovered area and the average POD (per subarea and total as seen in figure 2.6);
- Accept the solution if better then the last or apply the Metropolis criterion (Metropolis et al., 1953).

Check for termination

- Terminate when the temperature reaches a minimum value, when it converges or when a maximum number of iterations is achieved.

Mutation

- Select the subarea between a pair of tracks with lower coverage quality;
- Place two tracks closer together, achieving higher coverage in the subarea in between;
- The amplitude of the mutation Δ_i is obtained from equation 5.6, where r is a random number between -1 and 1, N is the neighbourhood size and T is the temperature of the process. It must be inferior to the previous track spacing s_i in subarea i;
- Update spacing s_i (equation 5.5);
- Calculate the contribution (weights w_j) of each subarea Δ_j ($j \neq i$) to compensate for Δ_i (equations 5.10 and 5.8);
- If there are no uncovered areas ($ucells = 0$) then maximize the average POD ($avgpod$) using a similar procedure where track spacing is decreased in the region where the average POD is lower; in this case the contribution of each region Δ_j depends directly on the average POD in each of those regions (equation 5.9).

$$s_i = s_i - \Delta_i \tag{5.5}$$

$$\Delta_i = rNT \tag{5.6}$$

$$\Delta_i = -\sum_{\substack{j=1 \\ j \neq i}}^{k} \Delta_j \tag{5.7}$$

$$w'_j = \frac{1}{ucells_j} \tag{5.8}$$

$$w'_j = avgpod_j \tag{5.9}$$

$$w_j = \frac{w'_j}{\sum_{\substack{j=1 \\ j \neq i}}^{k} w'_j} \tag{5.10}$$

$$\Delta_j = w_j \times \Delta_i \tag{5.11}$$

5.4.2 Strategy for the case with nadir gap

The most used technique to handle this problem is the uneven lawn-mowing coverage pattern, where consecutive pairs of tracks cover each other's nadir. Here, the distinction between odd and even areas is performed because they must be handled differently. Even areas will have smaller spacing since the adjacent tracks need to be closer together in order to cover their nadir gaps. Even areas contain the complete nadir region belonging to each of the adjacent tracks, even the portion that lies on the adjacent areas (the nadir extends to each side of the track considering that the sonar is being carried with zero roll angle). This strategy differs from the previous one on the mutation phase:

Mutation

- Similar to the previous one, except that adjusting odd subareas only affects other odd subareas, while adjusting even subareas only affects adjacent subareas;
- The purpose of controlling track spacing in even subareas is mainly to cover their nadir gaps, therefore only the adjacent subareas should compensate for the variation of track spacing;
- Even subareas should remain untouched when adjusting odd subareas so the nadir gaps remain covered.

5.5 Environment and path representation

It is assumed that some knowledge about the environment is available because it wouldn't make sense to optimize a path to an unknown type of terrain. Information about the topography of the survey area may be downloaded from a database or other available sources. For testing purposes, a fractal terrain generator was used. A regular grid is then created inside this area. Each cell holds

topographic information about the seafloor (longitude, latitude and depth) and a corresponding detection performance estimate that is obtained when a given solution is evaluated. Choosing the correct grid resolution is a critical task, since its manipulation is computationally intensive and it also affects the accuracy of the detection performance estimation.

Typically, coverage problems are solved by employing a path planning strategy that uses a set of equally spaced parallel tracks to cover the terrain. The path, composed by these tracks, is defined by a set of consecutive points which will always lie on the convex polygon that delimits the survey area. The location of these points is not related in any way to the location of the cells in the grid since these are only used to provide information about the seafloor and to estimate detection performance. It is not strictly required that tracks are parallel to each other but this usually happens because it makes it easier to generate a path that doesn't leave unexplored regions of space. Moreover, the success of the mission is highly dependent on the quality of the sonar acquired data. Typically a sonar does not work well when the vehicle is turning (the data gets distorted) so it is imperative that the path includes long straight tracks with minimal number of turns.

It is considered that a path can be fitted at constant depth or at constant altitude. A path with constant depth will not take into account the structure of the terrain, although it has to meet the spatial constraints defined by the environment. Such paths are more efficient in terms of required energy and time, but if the topography of the survey area is complex (rough instead of a smooth surface, with a high standard deviation on elevation) it may exhibit poor detection performance. On the other hand, a path defined to maintain a constant altitude from the bottom terrain can display a better overall detection performance (certainly decreasing its standard deviation) but it will require more time and energy to be executed by the vehicle since its length is bigger. Such a path can be seen as a vertical projection of the constant depth path on the bottom. The calculation of a constant altitude path involves dividing each (constant depth) track in smaller subtracks and fitting them to the terrain, between two specific horizontal positions, performing a simple linear regression in 2D space. The algorithm for fitting tracks to the bottom is as follows:

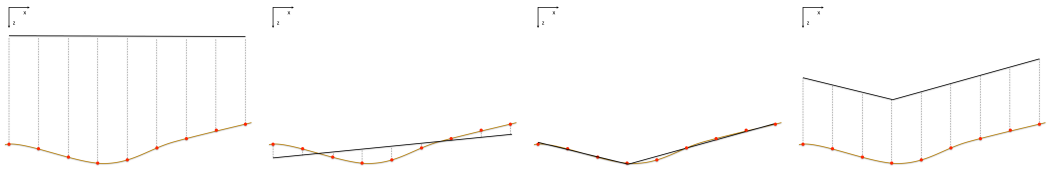
1. Select a track to be fitted;

2. Estimate the depth of several points along the vertical projection of track on the terrain; the distance between the points is a prespecified parameter;

3. Perform a linear regression to fit a straight line to the calculated positions on the terrain, maintaining the direction defined by the track;

4. Calculate the regression errors along the fitted track;

5. If the highest error is higher than a given prespecified maximum error, then divide the track at the corresponding position and repeat the procedure for each subtrack; track division leads to a reduction of the regression error;



(a) Depth estimation (b) Linear regression (c) Track division and new regression (d) Add offset representing altitude

Figure 5.2: Graphical demonstration of the track fitting procedure.

6. Add an offset to the fitted track to ensure that it is positioned above the surface of the terrain; ensure that its starting position is not at a higher depth than the ending position of the previous fitted track;

7. Add an offset corresponding to the desired altitude.

A graphical demonstration of this algorithm is presented in figure 5.2. Depth estimation is performed using Inverse Distance Weighting (Shepard, 1968) of the depth measurements surrounding the prediction locations.

5.6 Using infeasible data

Many real world problems are highly constrained which means that the regions containing feasible solutions are usually distributed in space and small. Some EAs regard infeasible solutions as useless individuals but various research showed that using infeasible individuals may affect and benefit the search process (Yu and Zhou, 2008). While keeping the search space small is a valid reason to discard infeasible solutions, it was reported that techniques that apply this action perform worse than techniques that allow these solutions to become involved in the evolutionary process (Coit and Smith, 1996). Yu (Yu and Zhou, 2008) presented a theoretical study that showed that including infeasible solutions could significantly affect the performance of EAs by accelerating the speed of convergence towards optimal solutions.

Here, a method is proposed that allows the use of infeasible individuals in the EA, taking them into account in the construction of the approximate model. This will help the exploration process in the boundary between feasible and infeasible regions. This allows the model to predict the location (feasible or infeasible region of the search space) of the individual being evaluated, which in turn will enable the EA to choose the best candidate individual to be a part in the next generation. In practice, a fourth output is added to the ANN model which describes the feasibility of a given solution and train the ANN with feasible and infeasible data. Therefore, when the performance of a solution is being estimated, its feasibility probability is being estimated as well. This information is then considered by the variation operators of the EA and will determine if the candidate solution is accepted for precise evaluation and consequent integration in the next generation population. The integration of such strategy creates a data management problem since

the entire search space needs to be sampled. To handle this issue, it was decided to sample each region of the search space independently, controlling the density of each dataset.

5.7 Data density reduction

An accurate approximation of the exact fitness function may be difficult to achieve when the amount of data available is inadequate, the data is not uniformly distributed and as a result not fully representative of the decision and objectives spaces. Generally speaking, the more accurate and the denser the training data is, the more accurate the approximate model will be. While a huge amount of the training data may guarantee generality of the model, it will require a very long training time. Improved efficiency can be achieved if redundant data is identified and eliminated from the dataset while maintaining generalization. The primary objective of a data reduction mechanism is to achieve an optimum balance between density of sampling and volume of data. Thus this will lead to lower training time and possibly better generalization.

Here, a method for selecting efficient training data is proposed. Training data will be gradually generated during execution of the algorithm and will be continuously used for the online training of the ANN. Data selection and management are critical in this case, as the amount data will become extremely large if it is continuously accumulated. Density is controlled by establishing a minimum distance between individuals. This distance is measured in the decision space using the Euclidean distance. That is, the distance between two individuals x and y is calculated as:

$$|i(x) - i(y)| = \sqrt{|i_t(x) - i_t(y)|^2 + |i_\theta(x) - i_\theta(y)|^2 + |i_d(x) - i_d(y)|^2 + |i_a(x) - i_a(y)|^2 + |i_v(x) - i_v(y)|^2} \quad (5.12)$$

where the parameters t, θ, d, a and v, that define the individual, are normalized. Different distances were considered in feasible and infeasible space in order to have a better understanding of the impact of this mechanism on the overall performance of the ANN. In the next section a series of experiments are performed to evaluate the behaviour of the adopted strategy.

5.8 Global mission planning

If an MCM operation involves multiple subareas, there is a need to build a path that efficiently interconnects all the subareas. The problem consists in choosing the best sequence of subareas to visit and the best internal path for each subarea. The optimization goal is to find the best global path that maximizes the global POD (calculated by performing an average of the POD of each subarea, considering its size) while minimizing the use of the vehicle's resources (time and energy). In practice, it is computationally expensive to find the optimal search path for the Travelling Salesman Problem (TSP) (Winston and Goldberg, 1987), so the algorithm should return a search path that is near-optimal. The calculation of the search path involves the computation

of a matrix of pairwise distances between exit and entry points to facilitate and speed up the optimization process.

The Iterated Local Search (ILS) algorithm (Stützle, 1999) was adopted to solve the global planning problem, for its simplicity and for the very good results it provided when applied to the TSP (Johnson and McGeoch, 1997). ILS consists in the iterative execution of a local search procedure to solutions that are obtained by perturbation of the previous local optimum. The local search algorithm is applied to obtain a new solution in the neighbourhood, which is then subjected to an acceptance criterion to decide which solution is the next locally optimal solution. The perturbation must be sufficiently strong to allow the local search to effectively escape from local optima and to explore different solutions, but also weak enough to prevent the algorithm from becoming a simple random restart algorithm. The initial solution is obtained using the nearest neighbour algorithm. The perturbation algorithm consists in a "double-bridge move" (4-opt exchange algorithm). It involves partitioning the current local optimal solution into 4 pieces (a,b,c,d) and putting it back together in a specific and mixed ordering (a,d,c,b).

Then, the iterative fast 2-opt exchange algorithm (Martin et al., 1991) is used as our local search procedure. The 2-opt algorithm consists of improving a tour until its length cannot be decreased by any 2-changes. If a 2-change leads to a decrease of the tour length, the exchange is executed and this requires inverting and re-writing part of the tour. To find a 2-change which decreases the tour length, all possible pairs of links are considered for a possible exchange. It does not make sense to consider pairs of links which are very far apart in space and this is the reason why the fast 2-opt algorithm was chosen. This method only takes into account k closest points when considering a given 2-change. The adopted acceptance criterion was the one used in the Large-Step Markov Chains approach (Martin et al., 1991, 1992). In particular, the new locally optimal solution is always accepted if it is better than the current one. Otherwise, if it is worse, it is accepted with a given probability.

In a typical search operation area, certain locations may have increased probability of having the search object or increased importance in the global context of the military operation. In this work, a priority based approach is adopted for CPP. Different priorities $P \in \{0, 1, ..., 9\}$ are assigned to different subareas within the target area based on a priori knowledge of the terrain. The AUV will visit the subareas by order of priority. As in planning without priorities, the goal is to find the path that guaranties a certain amount of detection performance while minimizing the use of resources (time and energy), but just a certain sequence of areas is considered by the acceptance criterion. It was decided not to allocate resources for each area proportionally to its priority level because it does not guarantee better overall performance. For example, an AUV covering an area with higher priority and characterized by having simpler topography and smaller currents will require fewer resources than covering an area in a more complex scenario (even if it has lower priority). It is assumed that the process of defining the priority classes is accomplished manually. It may be possible that some aspects of this process could be automated, but this is beyond the scope of this work.

5.9 Experiments

The goal of the following experiments is to:

- achieve a better understanding of the search capabilities of our algorithm;
- prove that our informed EA (EA whose variation operators use a ANN) is superior in terms of performance to the standard EA (initial adaptation from SPEA2);
- demonstrate that our multi-stage multi-objective algorithm can solve the mission planning problem successfully;
- show that the integration of our local search strategy increases the efficiency of the search process.

As the optimal Pareto front for our mission planning problem is not known, a procedure adopted by Diego et al. (Pinto and Barán, 2005) is used to obtain an approximation of the Pareto front for each algorithm and different sets of settings. The procedure consists of the following steps:

- each algorithm with a specific combination of settings is executed 10 times, obtaining 10 independent sets S_i of non-dominated solutions;
- these sets are combined and the non-dominated solutions are stored in a set S_f;
- the non-dominated solutions in S_f are used as an approximation of the true Pareto-optimal set.

The process of tuning an EA for a given application is complex because there is a large number of alternatives and very limited knowledge about the effect of the parameters on the algorithm's performance is available. There is still a lack of theoretically proven principles to help the user find good values for these parameters. The current opinion on EAs admits that specific problems require specific EA configurations for acceptable performance (Back et al., 1997). Therefore, an exhaustive analysis of the effects of the main parameters needs to be executed in order to understand their impact on the search process.

5.9.1 Tuning the standard EA

5.9.1.1 Population size

Here, the intention is to obtain a better understanding of the role that the population size plays in the performance of this EA. To this purpose, a number of experiments were performed consisting in running the EA using distinct populations of sizes, namely containing 50, 100 and 200 individuals. The results of the experiments are presented in the dominance table 5.1, where the different obtained Pareto fronts are compared by assessing the ratio (%) of the number of non-dominated

Dominance	**50**	**100**	**200**
50	-	1.49	2.41
100	28.13	-	4.82
200	43.75	19.40	-

Table 5.1: Dominance table comparing the performance of the algorithm with different maximum population sizes. The set on row i is said to dominate the set on column j by $k\%$, where k is the cell's value.

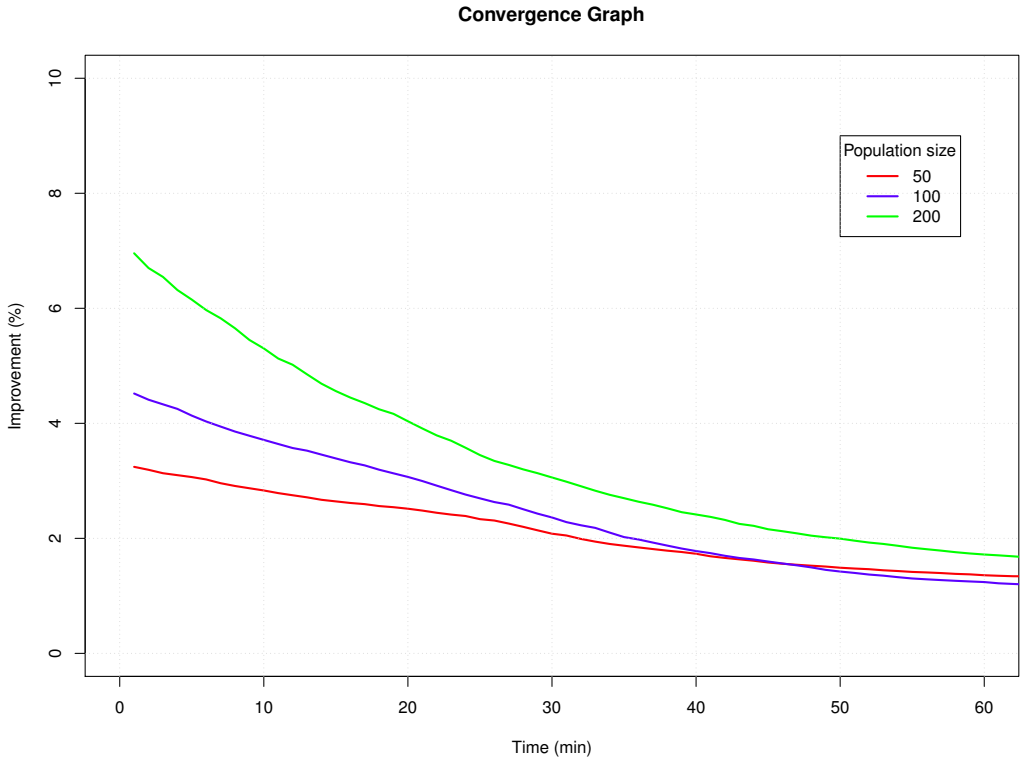


Figure 5.3: Improvement plots for the considered population sizes. Archive population stabilized with evolution as convergence was achieved.

solutions of distinct sets. A brief description of the obtained datasets is also shown in table 5.2. Figure 5.3 shows the evolution of the individuals in the archive during time.

Max population size	**Generations**	**Archive size**	**Volume**	**Uniformity**
50	475	64	0.04708	0.02746
100	421	67	0.07648	0.01811
200	327	83	0.07344	0.01392

Table 5.2: Description of the obtained dataset comparing the performance of the evolutionary process with different population sizes.

These results indicate that the performance of the EA improves by increasing population size. This is demonstrated by the growing dominance ratios and the faster convergence speed achieved by increasing this parameter. This improvement comes at the cost of a larger computational complexity, expressed by a decreasing number of generations in the same period of time. There is a trade-off between computational complexity and quality of the solutions. The EA with smaller population size has has reasonable computational complexity but is converging to poor solutions prematurely. Diversity is an important factor here because if no prior information about the best regions of the search space is available, then it is expected that the more diverse the initial population is, the greater the chance to find good solutions is. Therefore, the number of individuals in the population is an important parameter to control the amount of diversity. Our problem instance is characterized by a large decision space (many variables with a large range of admissible values) and in this case smaller population sizes proved to be insufficient because the individuals did not represent a large enough sample of the decision space.

Most of the performance gain is justified mainly by the use of a higher number of individuals in the initial population. Our elitist strategy, characterized by the use of an unrestricted external set with the best solutions to date, and our density control mechanism, that does not allow similar solutions in both the archive and the population, led to a particular situation where the maximum population size was never achieved (there was no truncation with the tested range of population sizes). Therefore, a higher population size in our case implies a higher diversity among the individuals in the population. A higher number of diverse individuals is potentiating exploration, more prominently in the beginning as shown in figure 5.3 where higher initial improvement ratios can be witnessed for the larger population sizes.

Table 5.2 also shows that the sets with higher population sizes, 100 and 200, had similar hypervolume metric values, both much higher then for the set with 50 individuals. This means that the obtained Pareto front is much wider in these cases but, as expected, the individuals in the archive of the largest set are more uniformly distributed since this archive also contains more individuals.

It is then clear that the tuning of the population size has significant influence in the efficiency and convergence of the EA.

5.9.1.2 Variation probabilities

Among the EA parameters, the recombination and mutation rates can be distinguished as critical parameters to its performance. Large values for the recombination rate introduces new individuals in the population faster. Similarly, if the mutation rate is too small, evolution would be difficult. However, a high mutation rate may cause the EA to become similar to a random search algorithm. Finding robust parameter settings that do not lead to premature convergence is not simple, as the impact of these settings on the EA's performance is complex. This said, simulations need to be performed in order to assess the influence in the search process and to find the suitable recombination and mutation rates for our algorithm. In this experiment, these rates are fine tuned by conducting a series of experiments using different combinations of these rates, seen on table

Set	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
P_R	0.2	0.2	0.2	0.2	0.5	0.5	0.5	0.5	0.7	0.7	0.7	0.7	1.0	1.0	1.0	1.0
P_M	0.2	0.5	0.7	1.0	0.2	0.5	0.7	1.0	0.2	0.5	0.7	1.0	0.2	0.5	0.7	1.0

Table 5.3: Sets of probabilities used for recombination and mutation.

5.3 and evaluating the impact on performance of the algorithm. Four different parameter values were tested for each rate, creating a total of 16 pairs of rates. The outcome of our simulations is presented in table 5.4. It was found that simulations with higher recombination rates performed better since, in general, sets with higher rate dominate more and are less dominated in average. The same conclusion can be taken for the mutation rate, although it is not so evident. In order to gain a better understanding of the behaviour of the algorithm with the different mutation rates, the improvement rates correspondent to the sets 13 to 16 (fixed recombination rate of 1.0) were ploted, as can be seen in figure 5.4. It shows that higher mutation rates lead to more stable rate of improvement (less fluctuations in evolution). The set 13 (in red) converged faster to sub-optimal solution as can be seen from the dominance table 5.4, indicating that the rate is too small. Sets 15 and 16, with mutation rates of 0.7 and 1.0, show a stable and sustained evolution even in the later stages of the simulation, indicating that the performance could be superior if the time allocated for the simulations was increased. Table 5.5 shows that runs with higher rates seem to populate archives with a higher number of individuals, occupying a bigger hypervolume and with better distributed individuals. Sets 12 and 16 are the best from the group. When comparing both sets, set 12 dominates twice the number of individuals that set 16 dominates. Set 12 has similar number of individuals in the archive to set 16, but these are better distributed. Considering these results, the best settings would be set 12 since the dominance aspect is considered to be more important then the uniformity of the individuals in objective space.

During design phase of this algorithm, using recombination or mutation probabilities (meaning we wanted to use all individuals for reproduction) was not considered despite the fact that they were used in most case studies found. We intended to use the recombination process to explore the search space and the mutation process solely to improve the known Pareto front. It is understandable that there may be a need to limit the exploration process but this does not apply to the exploitation process where we can produce a mutated individual that dominates the original. Also, these results show once more that the optimal combinations of rates are problem and algorithm dependent.

5.9.1.3 Mutation magnitude

The mutation magnitude parameter represents the maximum amount of change that is allowed in a single mutation operation. The actual mutation size will be defined by a uniformly distributed random number between zero and the product of the mutation magnitude parameter and the size of the variable range. A mutation with large magnitude is likely to produce large variations which

Dom	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	Avg_1	Avg_2
1	-	16.88	49.35	7.41	18.92	10.75	28.75	6.67	10.00	25.88	9.57	0.00	21.65	7.61	14.89	0.93	15.28	9.89
2	7.23	-	25.97	3.70	8.11	3.23	15.00	1.11	3.33	16.47	5.32	0.93	12.37	2.17	6.38	0.93	7.48	
3	1.20	3.90	-	2.47	4.05	1.08	0.00	0.00	1.11	2.35	2.13	0.00	1.03	2.17	0.00	0.00	1.43	
4	4.82	11.69	58.44	-	20.27	6.45	23.75	6.67	7.78	28.24	10.64	3.74	17.53	9.78	18.09	2.78	15.38	
5	0.00	10.39	27.27	3.70	-	3.23	13.75	2.22	2.22	11.76	3.19	0.00	7.22	4.35	3.19	0.00	6.17	11.71
6	10.84	22.08	63.64	6.17	17.57	-	33.75	8.89	7.78	42.35	10.64	2.80	17.53	7.61	13.83	3.70	17.95	
7	2.41	5.19	18.18	2.47	6.76	3.13	-	2.22	4.44	11.76	6.38	0.00	4.12	3.26	3.19	1.85	5.03	
8	8.43	18.18	55.84	7.41	21.62	7.29	38.75	-	7.78	31.76	17.02	0.93	18.56	11.96	17.02	2.78	17.69	
9	7.23	11.69	51.95	6.17	20.27	7.29	30.00	6.67	-	32.94	15.96	1.87	23.71	6.52	13.83	2.78	15.93	15.05
10	3.61	5.19	25.97	6.17	6.76	0.00	6.25	2.22	1.11	-	3.19	1.87	4.12	3.26	1.06	0.00	4.72	
11	1.20	9.09	28.57	4.94	16.22	2.08	20.00	3.33	6.67	20.00	-	1.87	10.31	3.26	9.57	0.93	9.20	
12	12.05	25.97	64.94	11.11	32.43	20.83	53.75	21.11	23.33	57.65	36.17	-	35.05	20.65	29.79	10.19	30.33	
13	2.41	5.19	20.78	4.94	6.76	4.17	13.75	2.22	8.89	17.65	7.45	0.00	-	2.17	4.26	0.93	6.77	14.50
14	1.20	14.29	46.75	4.94	13.51	4.17	23.75	5.56	3.33	25.88	10.64	0.00	17.53	-	13.83	0.93	12.42	
15	9.64	12.99	44.16	6.17	18.92	9.38	20.00	6.67	5.56	17.65	12.77	0.93	11.34	8.70	-	0.93	12.39	
16	13.25	28.57	70.13	7.41	29.73	10.42	56.25	20.00	15.56	49.41	25.53	4.67	23.71	17.39	24.47	-	26.43	
Avg_1	5.70	13.42	43.46	5.68	16.13	6.23	25.17	6.37	7.26	26.12	11.77	1.31	15.05	7.39	11.56	1.98		
Avg_2	17.07				13.47				11.61				8.99					

Table 5.4: Dominance table comparing the performance of the algorithm with different variation probabilities.

Set	**Generations**	**Archive size**	**Volume**	**Uniformity**
1	327	83	0.07344	0.01392
2	222	77	0.07722	0.01467
3	178	77	0.07464	0.01463
4	132	81	0.08037	0.01088
5	166	74	0.07273	0.01419
6	112	93	0.07896	0.01342
7	91	80	0.07586	0.01526
8	74	90	0.07270	0.01285
9	99	90	0.07112	0.01205
10	81	85	0.07678	0.01394
11	59	94	0.06640	0.01197
12	40	107	0.07813	0.01231
13	54	97	0.06774	0.01097
14	43	92	0.07421	0.01257
15	39	94	0.07553	0.01217
16	33	108	0.07778	0.01077

Table 5.5: Description of the obtained dataset comparing the performance of the evolutionary process with different variation probabilities.

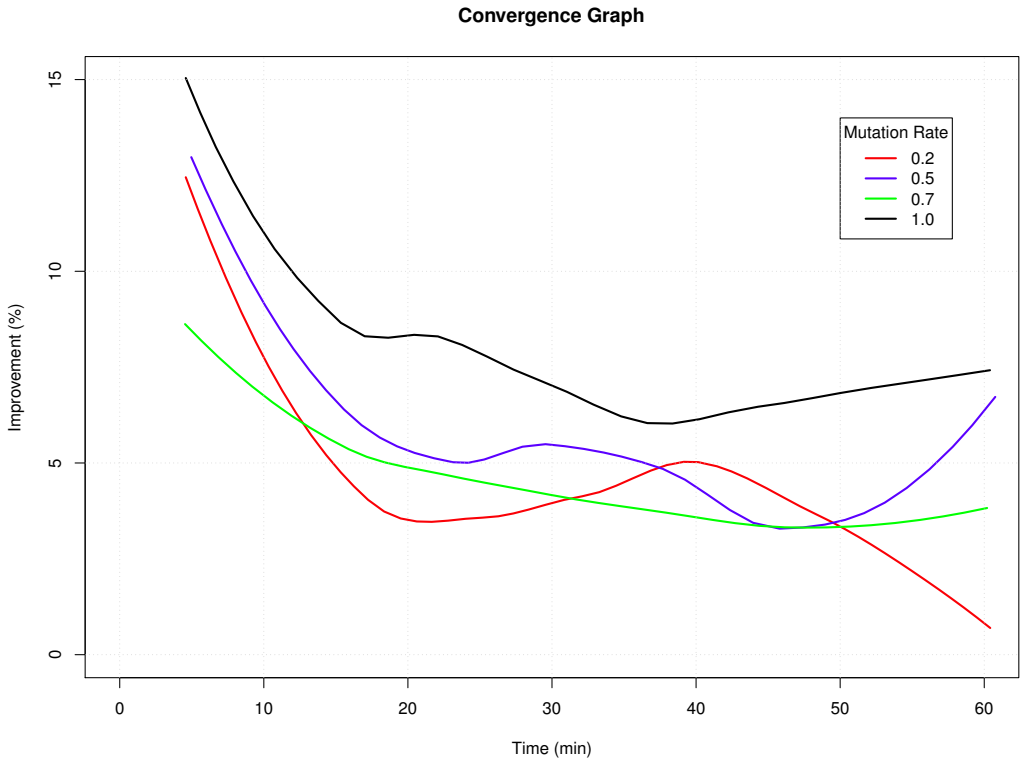


Figure 5.4: Improvement plots comparing the evolution using the different mutation rates with a recombination rate equal to 1.0. Archived population stabilized with evolution as convergence was achieved.

would facilitate better exploration of the undiscovered regions of the search space while a small magnitude usually produces small variations that are better for exploitation of the already found solutions. Whether potentiating exploitation or exploration would be the best objective for this operator is not clear, so it was decided to manually tune this parameter to have a better understanding of its impact on overall performance. The dominance table 5.7, obtained through experimentation with different magnitudes described in table 5.6, shows that the best performance was indeed obtained with the smaller mutation magnitude (0.1), as the set 1 presents higher average dominance rate and is less dominated by others.

The information provided by the convergence graph 5.5 helps justifying this best performance by displaying a much higher improvement ratio throughout the simulation run. It can clearly be seen that the improvement ratio follows a specific trend, increasing with decreasing magnitudes. The lack of convergence (none converged to zero improvement) indicates that better results could

Set	**1**	**2**	**3**	**4**
Mag	0.1	0.2	0.3	0.4

Table 5.6: Sets of mutation magnitudes using by the mutation operator.

Dom	**1**	**2**	**3**	**4**	**Avg**
1	-	11.22	9.57	37.65	15.71
2	1.96	-	5.22	36.47	11.46
3	5.88	7.48	-	40.48	13.73
4	1.96	0.00	0.87	-	0.98
Avg	6.13	9.35	8.26	41.00	

Table 5.7: Dominance table comparing the performance of the algorithm with different mutation magnitudes.

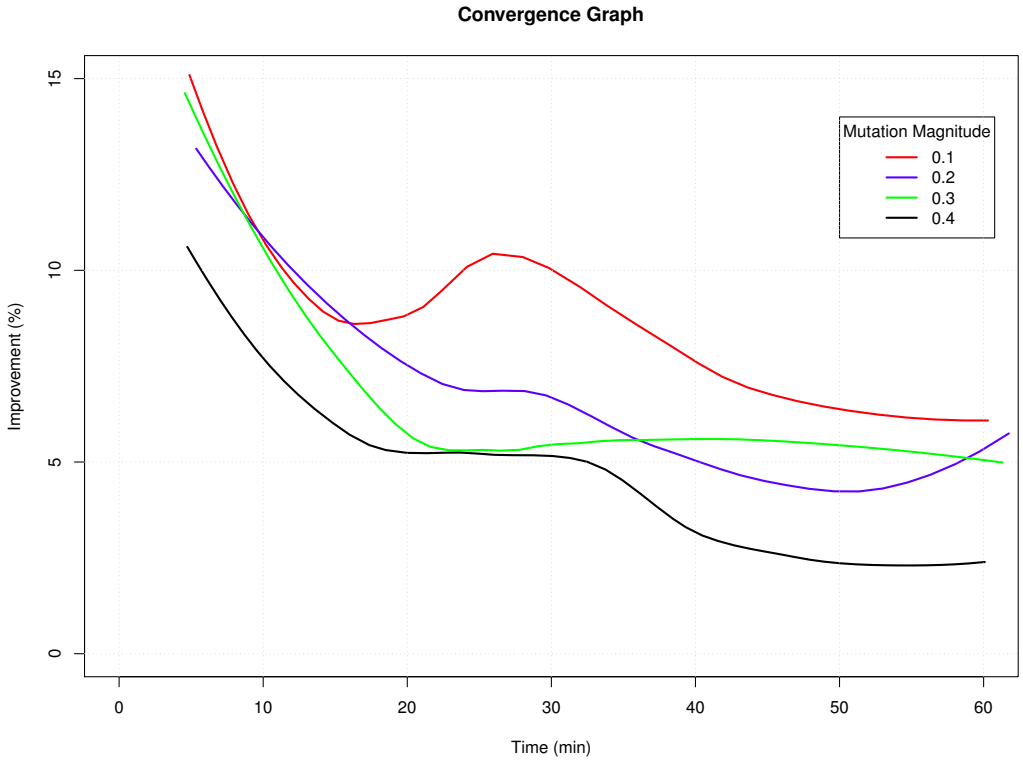


Figure 5.5: Improvement plots comparing the evolution using the different mutation magnitudes.

be achieved if more time was allocated for the simulation runs.

5.9.2 Tuning the informed EA

Here, the impact of some of the parameters that control the adopted fitness approximation strategy is going to be evaluated. These parameters affect the integration of the model in the EA and the level of approximation of the model.

Dom	1	5	10	20	Avg
1	-	20.00	12.12	28.41	20.18
5	12.73	-	26.26	38.64	25.88
10	4.04	18.89	-	18.18	13.70
20	14.29	8.89	10.10	-	11.09
Avg	10.35	15.93	16.16	28.41	

Table 5.8: Dominance table comparing the performance of the EA using different training generation gaps.

5.9.2.1 NN training generation gap

In order to integrate an approximate model in the EA, online training will need to be performed in order to control the quality of the model throughout the evolutionary process. The main reason that lead to the introduction of a ANN training gap is that the process of ANN training can allocate a large amount of time in the evolutionary process, so the impact of such operation on the performance of the EA needs to be assessed. Training control is usually performed once in a fixed number of generations. This method has the disadvantage of forcing a constant control frequency, meaning that the algorithm cannot react to the availability of recent training data that represents previously unknown regions of the search space. Since the training dataset size is expected to stabilize during the evolutionary process, the accuracy of the model should increase over time due to the consecutive training phases.

This lead us to perform a series of experiments comparing the performance of the informed EA using distinct training generation gaps (1, 5, 10 and 20 generations). Numerical results presented in table 5.8 show that the algorithm performed better using generation gaps of 1 and 5, with the expected drop-off in performance for higher values. The ideal training generation gap needs to be determined considering the specific problem instance being addressed and the chosen optimization algorithm because the training gap is dependent on the total number of generations (in our experiment the average was around 30 for 1 hour simulations) and on the number of generations needed to stabilize the training dataset (not achieved here).

5.9.2.2 Mutation Settings

The dominance table 5.10, obtained through experimentation with different settings described in table 5.9, shows that the best performance was obtained with the smaller mutation magnitude (0.1), as the set 1, 2 and 3 present higher average dominance rate and are less dominated then the others. As for the number of mutations that should be performed using the approximate model, the dominance table shows that the best results are achieved using 50. When using small mutation magnitudes there is not a real advantage in using a higher number of mutations since the individual's fitness values will be very close to each other when ranked.

Set	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Magnitude	0.1	0.1	0.1	0.2	0.2	0.2	0.3	0.3	0.3
N mutations	20	50	100	20	50	100	20	50	100

Table 5.9: Sets of mutation magnitudes and number of mutations used by the mutation operator.

5.9.2.3 Filtering in feasible and infeasible space

First, the performance of the training procedure needs to be investigated using the distinct data densities presented in table 5.11 in order to assess its impact on the performance of the EA. The sampling density experiments were performed on a continuously updated ANN training dataset where each data tuple was obtained through exact objective function evaluations.

As expected, better training performance was achieved using smaller distances between tuples in feasible space. Table 5.12 shows that the average training error, obtained through split sample validation, increased as the distance between tuples also increased. This suggested that the preferred normalized minimal distance between tuples in the feasible region should be 0.05 (or less). Comparing the sets with the same minimal distance in the feasible region reveals that the training error decreases if the minimal distance in the infeasible region is increased, reducing the total number of tuples in this region.

Table 5.13 presents the results of multiple simulations with the different trained ANNs. It is clear that the data density reduction mechanism (increased distance between data tuples) influences the EA: the average dominance of the obtained Pareto fronts decreased as data density decreased. The impact of the variability in ANN training performance observed earlier, in feasible and infeasible spaces, is also evident on the obtained results.

In any situation the created ANN was trained properly and was able to give reasonable estimations when fed with unknown inputs. Training data reduction handles data redundancy and improves data processing efficiency in terms of both storage and processing time. With this methodology, an efficient dataset can be maintained for online training of the ANN.

5.9.3 Adaptive tuning

The manually determined set of parameter values may not offer the best performance as they are kept constant throughout the evolutionary process. It was decided that the only parameters that should be dynamic are the mutation magnitude and the ANN generation gap. Variation rates should be kept constant since the informed algorithm will only generate individuals that are estimated to be non-dominated, hence guaranteeing that the evolutionary process is progressing. Our strategy consists in a decreasing variation of the mutation magnitude during the execution of the algorithm. The appropriateness of a small or large mutation magnitude changes dynamically depending on the state of the search process as well as the properties of the search space. A decreasing scheme was chosen because it is expected that a larger magnitude facilitates the

Dom	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	Avg_1	Avg_2
1	-	6.45	7.53	13.25	12.50	25.81	19.39	24.72	33.67	6.06	
2	22.33	-	12.90	13.25	22.34	24.73	26.53	28.09	36.73	9.09	20.73
3	11.65	6.45	-	10.84	11.46	23.66	26.53	31.46	33.67	8.08	
4	10.68	1.08	7.53	-	14.58	13.98	23.47	19.10	21.43	6.06	
5	11.65	7.53	11.83	14.46	-	18.28	21.43	28.09	27.55	2.02	14.27
6	10.68	2.15	5.38	7.23	14.58	-	21.43	10.47	29.59	4.04	
7	5.83	2.15	6.45	9.64	10.42	19.35	-	19.10	26.53	4.04	
8	8.74	6.45	7.53	4.82	9.38	9.68	18.37	-	17.35	6.06	10.12
9	7.77	7.53	3.23	6.02	8.33	11.83	13.27	24.72	-	7.07	
Avg_1	10.39	5.70	7.53	10.60	12.23	18.92	21.33	22.96	28.88		
Avg_2	7.87			13.92			24.39				

Table 5.10: Dominance table comparing the performance of the EA using different mutation settings.

Set		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Minimum Distance	**Feasible**	0.05	0.05	0.05	0.10	0.10	0.10	0.20	0.20	0.20
	Infeasible	0.05	0.10	0.20	0.10	0.20	0.40	0.20	0.40	0.80

Table 5.11: Sets of decision space distances used by the filtering mechanism.

Set	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Avg Tuples	746.6	609.3	702.5	346.4	313.9	268.9	113.7	89.5	77.6
Avg MSE	0.019	0.015	0.009	0.032	0.030	0.015	0.059	0.023	0.007

Table 5.12: ANN training performance using datasets with the distinct densities presented in table 5.11. The average number of tuples in the dataset during the execution of the EA is presented. Split sample validation was performed for assessing the training error, involving randomly splitting the dataset into two subsets, using one for training and the other for validating the generality of the result. The validation error is the one that is presented.

Dom	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	Avg_1	Avg_2
1	-	13.95	1.19	16.49	14.81	18.28	27.17	15.48	11.11	14.81	
2	17.78	-	0.00	21.65	14.81	20.43	27.17	11.90	12.12	15.73	21.63
3	27.78	37.21	-	36.08	28.70	35.48	46.74	34.52	28.28	34.35	
4	6.67	5.81	1.19	-	0.00	6.45	11.96	9.52	5.05	5.83	
5	21.11	24.42	5.95	17.53	-	13.98	22.83	25.00	11.11	17.74	12.09
6	13.33	18.60	2.38	16.49	4.63	-	20.65	15.48	10.10	12.71	
7	15.56	10.47	1.19	11.34	1.85	7.53	-	13.10	7.07	8.51	
8	16.67	17.44	4.76	20.62	18.52	22.58	22.83	-	17.17	17.57	12.69
9	15.56	12.79	2.38	17.53	7.41	13.98	11.96	14.29	-	11.99	
Avg_1	16.81	17.59	2.38	19.72	11.34	17.34	23.91	17.41	12.75		
Avg_2	12.26			16.13			18.03				

Table 5.13: Dominance table presenting a comparison of the different Pareto fronts obtained by using distinct ANNs.

exploration of the of the search space in the beginning and smaller magnitudes help the exploitation of the already determined solutions in the end. The objective is to maintain the improvement ratio above a prespecified minimum value until the end of the run, achieving a balance between exploitation with the exploration in the search space. If the improvement ratio, calculated at the end of each generation, falls below a minimum value, the mutation magnitude is multiplied by a constant (inferior to 1) similar to the cooling ratio in SA, decreasing its value. The previously determined magnitude will be used as initial value. Similarly, the ANN training generation gap should increase while the training error is kept close to the desired value.

The results of the experiment are presented in table 5.14. The population determined by running the informed EA with adaptive tuning is superior to the one determined by the manually tuned EA. The Pareto Front was wider, with more individuals and these were more uniformly distributed. This proves that an adaptive mutation magnitude helps in increasing the diversity which in turn helps balancing exploration and exploitation of the search space which finally helps in improving the solution quality and the convergence rate of the algorithm. The ANN training generation gap, which started as 1, remained the same throughout the execution of the algorithm as the training error was higher then the desired value. The time that was allocated for this experiment was not enough to observe a variation of this parameter. Nevertheless, it can be concluded that the impact of dynamically controlling this parameter is small since the training algorithm detects error convergence and terminates the procedure without significant loss of time.

5.9.4 Standard EA VS informed EA

Since the EAs have a different structure (the inclusion of the approximate model implies the use of different variation operators and an additional phase to build and maintain the model) it was decided to analyse their performance at specific instants of time. Table 5.15 shows the results of the simulation. It can be observed that at the initial stages the Pareto set obtained by the informed

Tuning	Manual	Adaptive
% Dominated	25.42	4.17
Individuals	59	96
Generations	31	37
Volume	0.07198	0.07554
Uniformity	0.02003	0.01098

Table 5.14: Comparison of the obtained non-dominated sets using different tuning procedures.

EA is dominated to a larger extent by the one obtained by the standard EA. This can be due to the longer initialization phase that is executed on the former, as the neural network needs to be trained, while the latter starts the evolutionary process sooner. The increasing dominance that the Pareto set generated by the informed EA exhibits is a consequence of an improving representation of the search space, maintained by the ANN training dataset, as the search progresses. This proves that the integration of the ANN is advantageous to the search process since it is guiding the search to more promising regions.

5.9.5 Offline mission planning with EAs

It is important to exemplify what will be the typical output of the execution of our planning algorithm and demonstrate the role that the decision maker will have to play. For this test, a smooth terrain was randomly generated using a fractal algorithm that's popular for generating realistic looking terrain. It was decided to stop the algorithm after 1 hour so its behaviour can be analysed in detail. Our planning problem was subject to the constraints detailed in table 5.16. Five distinct solutions were selected from the Pareto front shown in figure 5.6 and these are fully described in table 5.17. They were chosen because they clearly demonstrate the trade-offs between the objective function values obtained in our mission planning procedure. It can be observed that a higher detection performance implies more time to complete the mission. As expected, it is also shown that the detection performance is better when using smaller inter track distances (a higher number of tracks is used). The velocity also has considerable impact on performance. This can be deduced because the best solutions (in terms of detection performance) use the minimum values for

Time (min)	10		20		40		60	
EA	**Std**	**Inf**	**Std**	**Inf**	**Std**	**Inf**	**Std**	**Inf**
% Dominated	2.70	26.67	10.11	18.60	23.71	12.50	30.39	4.76
Individuals	74	75	89	86	97	80	102	84
Generations	6	4	15	14	26	31	37	50
Volume	0.06510	0.07065	0.06787	0.07186	0.07241	0.07832	0.07780	0.07369
Uniformity	0.01499	0.01392	0.01336	0.01322	0.01145	0.01593	0.01255	0.01441

Table 5.15: Comparison of the obtained non-dominated sets as the EAs are executed.

Parameter	Bounds	Type of constraint
Battery capacity	800 Wh	vehicle
Velocity relative to water	$1 \leq v \leq 3$	vehicle
Sonar maximum range	80 m	sensor
Sonar vertical beam angle	$10 \leq \alpha \leq 70$	sensor
Maximum operating time	10 h	mission

Table 5.16: Design constraints used in this test.

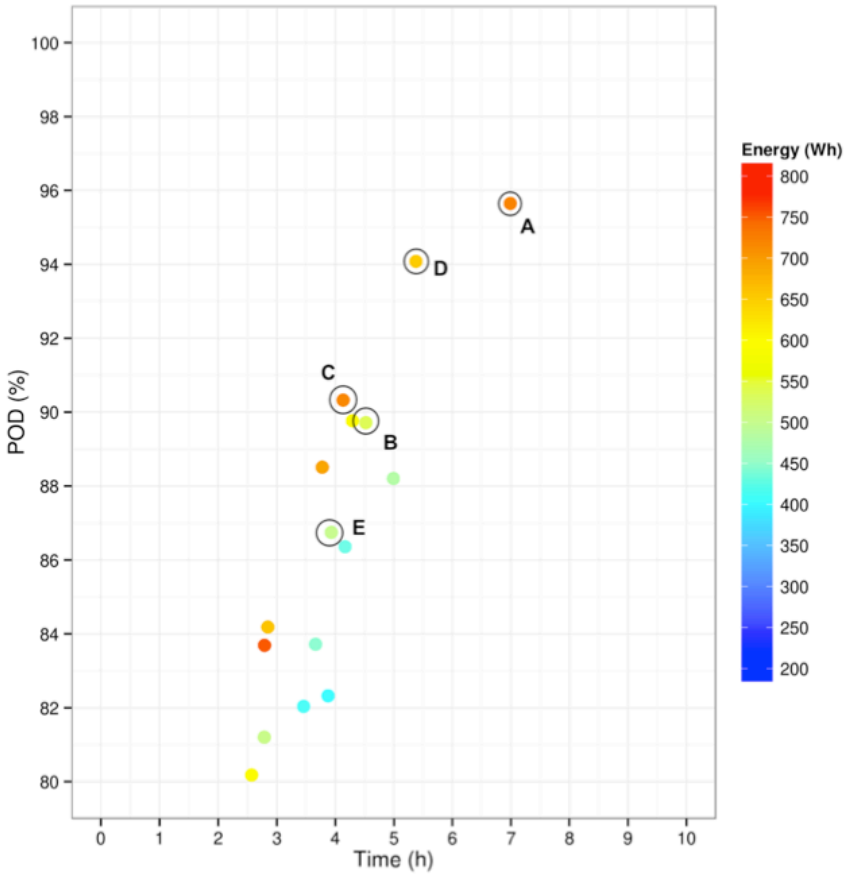


Figure 5.6: Truncated Pareto Front containing several solutions for the mission planning problem.

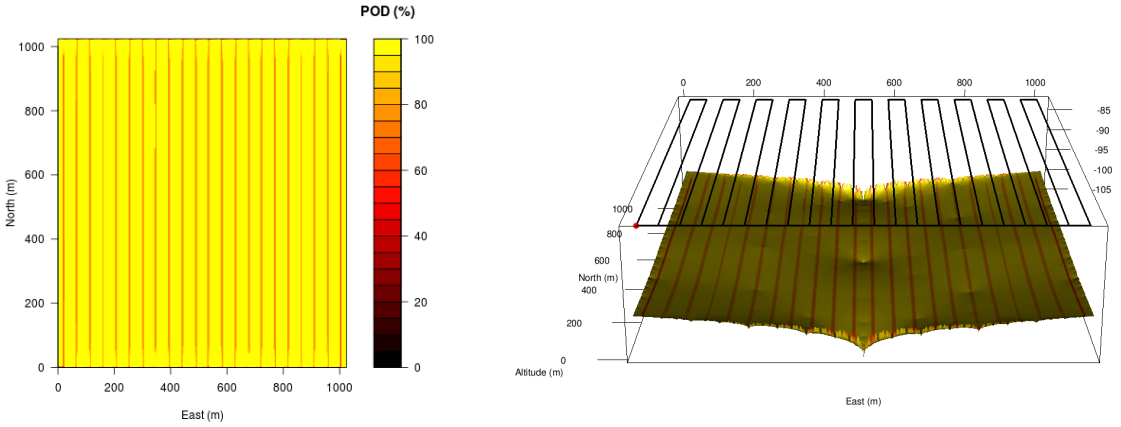
this variable as the energy that is saved by doing so is being used to increase the length of the path by using a higher number of tracks.

5.9.6 Local optimization

Non-dominated solutions previously found by our informed EA were used as initial solutions here. Table 5.18 presents descriptive statistics for the optimized set of solutions for covering a given area with a SSS. It shows the variation in the amount of insufficiently covered cells ("Min-Ratio") and the variation in POD. In average, a decrease of roughly 5% was obtained in the former

s	$\overline{POD}$ (%)	Energy (Wh)	Time (h:m:s)	Depth (m)	Altitude (m)	Spacing (m)	Velocity (m/s)	Direction (°)
A	95.65	728.80	6:59:20	-	13.24	41.13	1.07	91.44
B	89.72	537.04	4:31:11	-	11.63	63.26	1.14	32.31
C	90.33	723.73	4:07:59	82.38	-	57.46	1.34	259.87
D	94.08	656.93	5:22:51	81.24	-	50.96	1.15	16.70
E	86.75	500.53	3:55:52	-	18.00	72.31	1.18	201.79

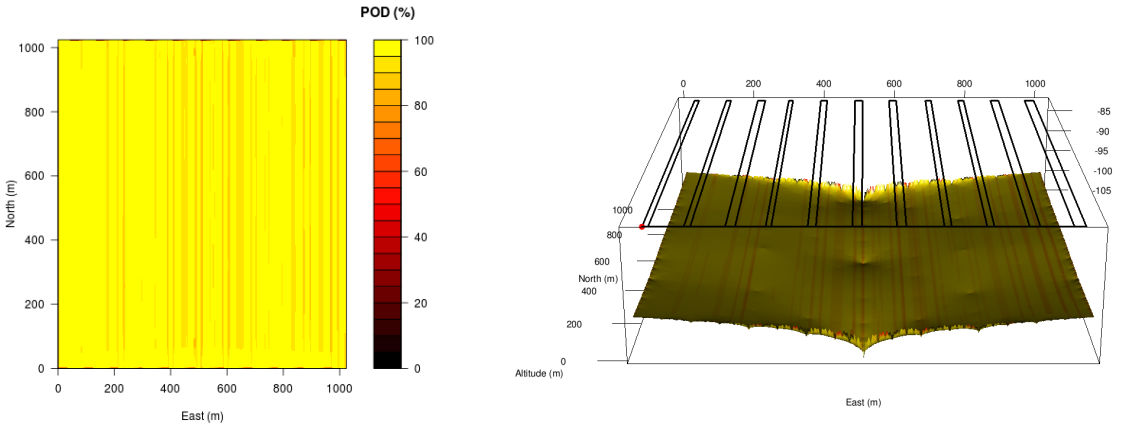
Table 5.17: Some distinct solutions in the Pareto Front.



(a) 2D coverage plot of original solution.

(b) 3D coverage plot of original solution.

Figure 5.7: Original solution obtained by the EA using a sidescan sonar.



(a) 2D coverage plot of optimized solution.

(b) 3D coverage plot of optimized solution.

Figure 5.8: 3D local optimization of the previous solution obtained by the EA using a sidescan sonar.

	Δ_{MR} (%)	Δ_{POD} (%)	**Time (s)**
Min	-19.41	-8.84	24.00
1Q	-10.86	-0.01	29.75
Mean	-4.99	0.59	64.74
3Q	-0.73	1.49	51.75
Max	11.81	5.51	380.00
SD	5.80	1.76	67.04
Var	33.64	3.10	4494.34

Table 5.18: Descriptive statistics of the complete set of solutions obtained for the coverage problem using a sidescan sonar.

and a increase of 0.6% in the latter, clearly showing the improvement made by local search in this scenario. This demonstrates the need for uneven lawn-mowing coverage pattern when using a sidescan sonar. The obtained results allowed us to take some additional conclusions related to the usefulness of this search process. As explained earlier, it is required an even number of tracks to cover the nadir gaps caused by the use of a sidescan sonar. If the original number of tracks is odd, then a track needs to be added or removed in order to use our algorithm. The simulations showed that while the performance is positively affected by an addition of a track, it is negatively affected by its removal. The local optimization algorithm could not compensate the decrease in performance caused by the removal of a track. Table 5.19 presents the descriptive statistics for the optimized set of solutions excluding the ones where tracks were removed. It can be seen that the average POD increases and that the amount of cells with insufficient coverage is reduced. In this scenario the preferred action is to add a track, but since mission planing constraints need to be respected, it may not be possible to do so. Therefore it can be concluded that, in a time critical application such as the one being addressed here, it may be better to simply skip the local optimization of solutions that required removal of a track since it is not worth the extra computational time. Figure 5.8 shows a graphical representation of a given solution before and after the local optimization procedure was executed. The improvement in this case was a decrease of 14.0% in the amount of uncovered cells and an increase of 0.9% in the average POD. This is visually identified by the almost inexistence of red regions in the optimized solution plot. Notice that the tracks were simply rearranged maintaining its direction and depth and that no tracks were added or removed. This demonstrates the usefulness and complementarity of both our local optimization algorithm and our EA.

Similar experiments were performed, but using a multibeam instead of SSS. The improvements are considered negligible as they are smaller by a factor of 5 then in the previous scenario, meaning that the gains do not justify the extra processing time.

	Δ_{MR} (%)	Δ_{POD} (%)	Time (s)
Min	-19.41	-1.99	24.00
1Q	-11.68	0.01	29.50
Mean	-5.43	0.81	64.77
3Q	-0.74	1.54	50.50
Max	2.09	5.51	380.00
SD	5.56	1.28	68.58
Var	30.85	1.64	4702.95

Table 5.19: Descriptive statistics of the set of solutions excluding the ones where tracks were removed.

5.9.7 Comparing informed EA with other mission planners

Three mission planners (Fang and Anstee, 2010, Stack and Smith, 2003, Williams, 2010) were selected, as they are suited to a search problem with similar assumptions to ours. A detailed description of the features of each algorithm is presented in table 5.20, but this study should be complemented with a few experiments to prove the superiority of our algorithm (Informed EA). Since their primary concern is to calculate track spacing between parallel tracks, those algorithms need to be fed with information calculated by our algorithm such as ideal vehicle velocity, coverage depth and direction. Therefore, the comparison made here is not fair since our algorithm needed a higher amount of time to calculate these parameters. However, this is a distinctive feature of our algorithm and even if this information is shared with the other algorithms, ours will still outperform theirs. Our experimentations included two different terrains, one is a simple planar terrain with a 2% slope and the other is a terrain with a complex (irregular) topography. The first experiment consisted of a comparison of all mission planners for local mission planning (planning for a single area). The results are presented in tables 5.21 and 5.22. In both scenarios our algorithm displayed the best estimated detection performance. Note that it was the only algorithm that consistently obtained solutions with estimated detection performance above the 95% prespecified minimum. Our algorithm has a high execution time for two specific reasons. First, because it is a complex optimization technique that estimates several mission parameters. Second, because the path fitting algorithm that creates a constant altitude path is computationally expensive. The other mission planners do not have this capability.

In the second experiment the performance of our algorithm and the algorithm developed by Fang et al. (Fang and Anstee, 2010) for global mission planning (planning for multiple areas) was compared. The results are presented in table 5.23. Multiple operating subareas where obtained through convex decomposition of a concave area. Then local planners were executed for each subarea and the global planner tested different combinations of the obtained solutions. Again, our algorithm found the solution with the best estimated detection performance but at the cost of a higher processing time. This is expected since our algorithm optimizes a number of variables, while the other just uses fixed track spacing and other prespecified parameters values. Our global

Planner	Features
Stack et al. (Stack and Smith, 2003)	• Local planner, considers single convex areas • 2D algorithm • Only calculates track spacing • Minimizes the amount of unsearched area that can be occupied by undetected mines • Assumes a perfect probability of detection • Does not consider SSS (nadir gap)
Fang et al. (Fang and Anstee, 2010)	• Global planner: calculates path for each subarea and interconnects them (solving the TSP with a GA) • 2.5D algorithm • Only considers track spacing • Does not optimize POD (fixed track spacing) • Minimizes global path length • Does not account for complex topography • Assumes the sensor is a SSS (uses uneven lawn-mowing but does not optimize track spacing) • Considers different shapes for the operating area and decomposes it in many subareas
Williams et al. (Williams, 2010)	• Local planner, considers single convex areas • 2D algorithm • Only considers track spacing • Maximizes the reward obtained from performing a track • Iteratively adds random tracks maximizing the benefit in terms of average POD increase • The POD is a function of range and seabed type • Does not account for complex topography • Assumes the sensor is a SSS (optimizes track spacing and accounts for nadir gap)

Table 5.20: Comparison of other techniques also used for mission planning. These techniques are described on section 2.2

.

Planner	$\overline{POD}$ (%)	Energy (Wh)	Time (h:m:s)	Evaluation Time (m:s)
Informed EA	95.775	521.612	03:19:48	01:47
Stack	87.989	605.420	04:00:53	00:48
Fang	86.274	619.521	04:11:51	00:03
Williams	89.978	594.202	03:47:29	01:10

Table 5.21: Obtained solutions using different local planners on simple terrain.

Planner	$\overline{POD}$ (%)	Energy (Wh)	Time (h:m:s)	Evaluation Time (m:s)
Informed EA	98.226	547.486	04:27:25	00:54
Stack	85.827	519.495	04:52:49	00:02
Fang	94.223	472.692	05:12:20	00:05
Williams	96.081	480.205	03:54:22	02:17

Table 5.22: Obtained solutions using different local planners on complex terrain.

Planner	$\overline{POD}$ (%)	Energy (Wh)	Time (h:m:s)	Evaluation Time (m:s)
Informed EA	92.285	344.125	01:43:07	00:39
Fang	86.280	158.195	01:44:49	00:04

Table 5.23: Obtained solutions using different global planners on complex terrain and area boundary.

planning procedure also takes more time as it needs to test a much higher number of combinations of solutions generated by the local planning procedure, while the other just test different entry and exit points relative to one solution in each subarea. These experiments prove that our informed mission planner is more complete, although at the cost of higher complexity and higher execution time.

5.9.8 Comparison with geometrical planner

The final assessment is demonstrating where the advantages of the evolutionary planner are compared to the classic geometric planner. The basic principle of the geometric path planning approach is to calculate the position of the parallel tracks by using the effective search width and the height that enables maximum detection performance at maximum viewing angle. Thus, if considering a planar terrain, this method is exact and delivers the shortest path with maximum detection performance. But, as the topography becomes more complex, the algorithm will lose its optimality due to the errors introduced and its performance will become worse than the evolutionary planner's performance. Additionally, a geometric path however does not specify how fast a robot should traverse the path, so information about time and energy consumption are not considered in path planning.

For the purpose of demonstrating the benefits of each algorithm, simulations will be performed using a sloped planar terrain and multiple complex terrains, with increasing standard deviation, generated with a fractal terrain generator.

5.9.8.1 Sloped planar terrain

Planner	$\overline{POD}$ (%)	Energy (Wh)	Time (s)	Length (m)	Direction (^{o})	Evaluation Time (s)	Fitting method
Geometric EA	100.0	22.4	823	893.7	0	4	altitude
Evolutionary (1^{st} generation)	97.1	24.8	731	575.3	1.4	17	altitude
Evolutionary (10^{th} generation)	97.9	19.0	888	819.6	289.0	60	depth

Table 5.24: Obtained solutions using different planners on sloped planar terrain (1.15 degrees facing north).

While both algorithms cover the complete terrain, they generate solutions with different characteristics. As expected for planar terrains, the geometric planner produced a solution that achieves maximum detection performance. The evolutionary solution was able to converge to a solution early. Between the first and tenth generations, the algorithm was able to improve POD and decrease time and energy requirements. However this came at the cost of inferior detection performance. This feature of the algorithm exists to avoid wasting a large amount of resources on

marginal improvements on detection performance. So, as soon as the complete terrain is covered and the specified minimum detection performance is achieved, the algorithm terminates. Once again, this is the expected behaviour for planar terrains. Lets see what happens when the roughness of the terrain is increased to match real world environments.

5.9.8.2 Fractal terrain generation

Fractal based models are considered to be an efficient method for creating realistic-appearing terrain (Fournier et al., 1982). Using the diamond-square algorithm (Miller, 1986), a fractal surface is generated and used to model the terrain.

SD	**Pln.**	$\overline{POD}$ **(%)**	**Uncovered (Area m^2)**	**Energy (Wh)**	**Time (s)**	**Length (m)**	**Evaluation Time (s)**
0	G	(Planned) 100.0 (Real) 100.0	(Planned) 0 (Real) 0	465.2	18608	18608	2
0	E	99.2	0	533.2	17261	19093	295
2	G	(Planned) 100.0 (Real) 99.9	(Planned) 0 (Real) 0	514.1	20562	20562	2
2	E	99.2	0	508.33	21647	20929	301
4	G	(Planned) 100 (Real) 98.7	(Planned) 0 (Real) 352	483.4	19335	19335	2
4	E	99.2	0	533.4	21344	21344	307
6	G	(Planned) 100 (Real) 99.1	(Planned) 0 (Real) 80	590.9	23637	23637	2
6	E	99.2	0	502.4	25540	22449	296
10	G	(Planned) 100 (Real) 99.0	(Planned) 0 (Real) 288	594.2	23766	23766	2
10	E	99.2	0	619.7	29895	29895	291
16	G	(Planned) 100 (Real) 90.0	(Planned) 0 (Real) 94560	595.0	23768	23768	2
16	E	98.5	0	510.4	37476	25483	291

Table 5.25: Performance comparison between the geometric (G) and the evolutionary planner (E) using randomly generated terrains with average depth of 20 meters and SD standard deviation. The standard deviation was slowly increased to evaluate the impact on the performance of the algorithms. The evolutionary algorithm allowed the execution of 5 iterations.

Table 5.25 holds the results from the experiment, where the standard deviation parameter (given as input to the fractal algorithm for random terrain generation) was slowly increased. In all cases both algorithms were able to deliver valid solutions, although with varying levels of performance. Before analysing the results, it is important to understand the basic process for planning with the geometric planner and then these results will make perfect sense. In order to use an exact planning procedure without error, a planar terrain is required. By increasing the standard deviation that will be used by the terrain generator, the complexity of the terrain or its

roughness is being increased as a consequence. Thus, to apply the geometric planning technique a plane needs to be fitted to the 3D terrain, creating an approximate planar image of that terrain. This approximation will lead to planning errors as a consequence. These results clearly show this behaviour of the geometric planner, where the detection performance decreased and the amount of uncovered terrain increased as roughness was increased. It is undeniable that the geometric algorithm has very good performance in simple environments. It may be usable for quick offline planning but lacks flexibility for online planning by not controlling the amount of resources spent by the vehicle during the mission. Nevertheless, it is a very good candidate, as initial solution, to be fed to the evolutionary planner during initialization to ensure that a solution close to optimal (in terms of path length and detection performance) will be found.

Chapter 6

Online mission planning

6.1 Introduction

As mentioned previously, offline CPP algorithms described in literature assume that the map used to plan the path accurately represents the environment and that the vehicle will be able to precisely execute the path, ignoring any uncertainty in sensing or control. However, these assumptions may not hold when operating in a challenging environment, where the robot's position estimation is subject to error, perturbations such as currents affect its control actions and the acoustic sensors used to map the environment might detect inaccuracies in the prior map.

To be able to conduct AUV coverage tasks in such environments, a novel 3D coverage path replanning method[1] is presented that does not rely on the unrealistic assumption of an idealized path execution (see next section). In this section, the application of the EA (explained in the previous section) to a partially known environment, while the AUV is executing a mission, is discussed. From the online path planning perspective, environmental changes occur due to either update of mapping data by newly acquired information or due to the (erroneous) movement of the AUV. The environmental changes may lead to changing the current path in order to avoid collision or improve the path with respect to an optimization criterion in a new environment. Thus, two issues are distinguished regarding how to update the prior map and how to update a population of individual solutions while the AUV is moving. The approach presented here differs from other CPP methods in that paths for coverage are generated based on a coverage map that is actively maintained as the vehicle executed its mission. This allows paths to be generated online and can result in path planning methods that are adaptive to sensor data acquired in situ during the mission.

An idea that is particularly interesting to us is that problems seldom exist in isolation. A truly efficient system must expect to tackle related problems over its lifetime and it would be an advantage if such a system could improve its performance with experience. Such a learning system requires memory, a place for storing past experiences and intelligence that allows the use of that information to guide future operations. Clearly, a disadvantage of a system without memory is that it is forced to start from scratch when trying to solve every given problem. Then, a system

[1] This work has been submitted for publication (Abreu and Matos, 2017).

that uses a case base as a long term knowledge store in a new planning algorithm is proposed. Our approach borrows ideas from CBR in which old problem and solution information, stored as cases in a case-base, helps solve a new problem. One or multiple databases, or casebases to be exact, of problems and their solutions provides the online replanner with long term memory. The case-base does what it is best at: memory organization. The replanner, which coordinates the execution of local optimization or evolutionary search (needed in the worst case where no feasible solutions are retrieved), handles what it is best at: adaptation. The resulting combination takes advantage of both paradigms where our evolutionary approach in conjunction with ANNs, presented earlier, delivers robustness and adaptive learning while the case-based component speeds up the replanning process. In the approach described here, the problem is to adapt the current mission to a changing environment while it is being executed. The solution to the problem is the path that is found in the casebase and traversed. The casebase stores the best individuals, the Pareto set, found using the offline evolutionary planner. Hence, each case represents mission parameters such as track spacing, track direction, depth or altitude of the path relative to the seafloor and the vehicle velocity and corresponding expected mission performance according to pre-existent assumptions (prior map, ocean currents, energy available, etc). This CBR mission replanner is characterized not only by retrieval and repair of past routes but also by synthesis of new mission parameters whenever acceptable routes could not be retrieved. Thus the planner is capable of handling situations where an appropriate matching past cases are not found in the casebase. In such cases, it generates a new mission plan by:

- locally optimizing the current one;
- evaluating the cases in the casebase according to the new acquired information and choose the best;
- when no acceptable mission plan can be retrieved, executing the evolutionary planner to perform exhaustive search until one is found.

This ability to fall back from plan adaptation to plan synthesis is required to optimize performance and reduce excessive tweaking of old cases (higher processing time).

6.2 Mission overview

Before deriving a strategy for mission planning using an AUV for search tasks, lets have a look at a typical operational scenario. The AUV will have to follow a preplanned mission plan, avoiding detected obstacles if needed, sampling and analysing data and replanning to improve performance.

The mission planning system to be developed should consider the following tasks:

1. Mission (re)planning and control;
2. Navigation around the environment, covering the seafloor, according to a given mission plan and gathering data;

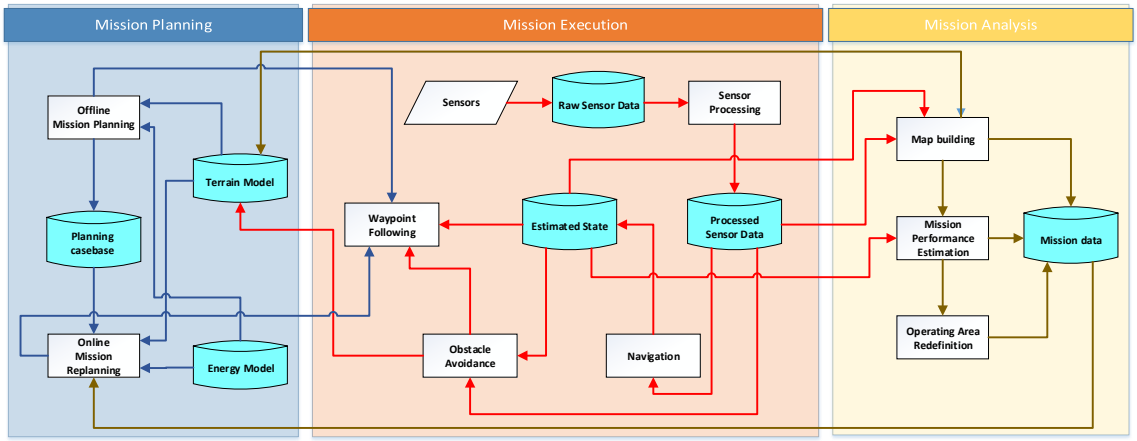


Figure 6.1: Mission flowchart.

3. Automatic analysis of the acquired data using onboard systems (determine if replanning the mission is needed);

4. Data upload to ground station or GPS update to reduce navigation system uncertainty (both need the vehicle to return to surface); After a given time the AUV may need to secure the data or reduce localization uncertainty, therefore it should return to surface to receive GPS fixes and/or communicate with ground station;

5. Obstacles detection and avoidance while executing mission.

The planner should monitor the waypoint-following procedure considering the current mission plan, the vehicle's resource usage and the accuracy of the navigation system. Specific events may trigger the analysis of the acquired data such as, for example, the arrival to a given waypoint, errors in navigation data, reaching a specific amount of available resources on the vehicle and so on. This event-driven approach is particularly applicable to the case when a vehicle is receiving intermittent position updates such as an AUV surfacing for GPS fixes. Each time the vehicle receives a update, the past trajectory is rectified and the coverage map is recalculated based on the new information. The sensor data from imaging sensors is processed to obtain a coherent map of the environment taking into account the sensor noise and uncertainty in position. For this reason the planner also keeps track of additional data such as the history of the AUV's position. The position accuracy defined by mission specifications governs how often the vehicle needs to surface for GPS updates. The operational endurance and range is dictated by vehicle specifications (batteries capacity, communication range if online communication is desired, storage capacity if data logging is desired, etc.).

Clearly, two distinct phases can be identified:

- Observation phase: waypoint following and obstacle avoidance;

- Calculation phase: Data processing and mission replanning.

These tasks may or may not be executed in parallel, but the planning system needs to execute both until the mission is executed successfully.

For the application scenario considered in this work, the following requirements have been determined for online path planning:

- Path search time vs. path optimality: the planner must quickly provide a path whenever replanning is necessary; there is a time deadline to complete the mission and limited vehicle's resources so the amount of time and battery capacity available need to be considered in the calculation phase; thus, it is acceptable for the planner to provide sub optimal paths in order to save runtime and/or make better use of the available energy;

- Multiple area global planning: the path planner must be able to handle a sequence of more than one operating areas; during mission execution new areas that require better coverage may be identified; it should use all the available knowledge at that time;

- Exploration characteristics: the planner must deal with an environment that may be different to the one previously considered;

6.3 Algorithm

6.3.1 Trajectory estimation

A good estimate of the complete vehicle trajectory is required in order to obtain a reliable estimate of mission performance. Usually for an autonomous system, a time-stamped geographical coordinate can be logged at least every second. Unfortunately, this leads to accumulation of a large amount of data that costs a lot of processing time when analysing the trajectory. To address this issue, trajectory compression strategies (based on the shape of a trajectory) are used, aiming to reduce the amount of points while not compromising much precision in the resulting data representation (Lee and Krumm, 2011).

Offline compression reduces the size of trajectory after the trajectory has been fully generated. Given a trajectory that consists of a series of time-stamped points, an offline compression algorithm generates an approximated trajectory by discarding some points with a negligible error from the original trajectory. Ramer et al. (Douglas and Peucker, 1973) developed the Ramer-Douglas-Peucker (RDP) algorithm for reducing the number of coordinate points required to represent a curve. This algorithm relies on a distance metric, ε, which establishes whether or not one of the original points in the curve will be included in the compressed representation. The value ε represents the perpendicular distance from a line segment in the trajectory to another point in the trajectory preservation. If the perpendicular distance is less than ε, then the point is kept and included in the result. The value ε can also be interpreted as the "granularity" of the resulting simplification. By changing this value, the resulting trajectory can be adjusted to be a "fine grained" or a "course grained" representation of the original trajectory. The smaller the value of ε, the closer

the reduced point trajectory will be to the original trajectory and in turn, more points from the original trajectory will be included.

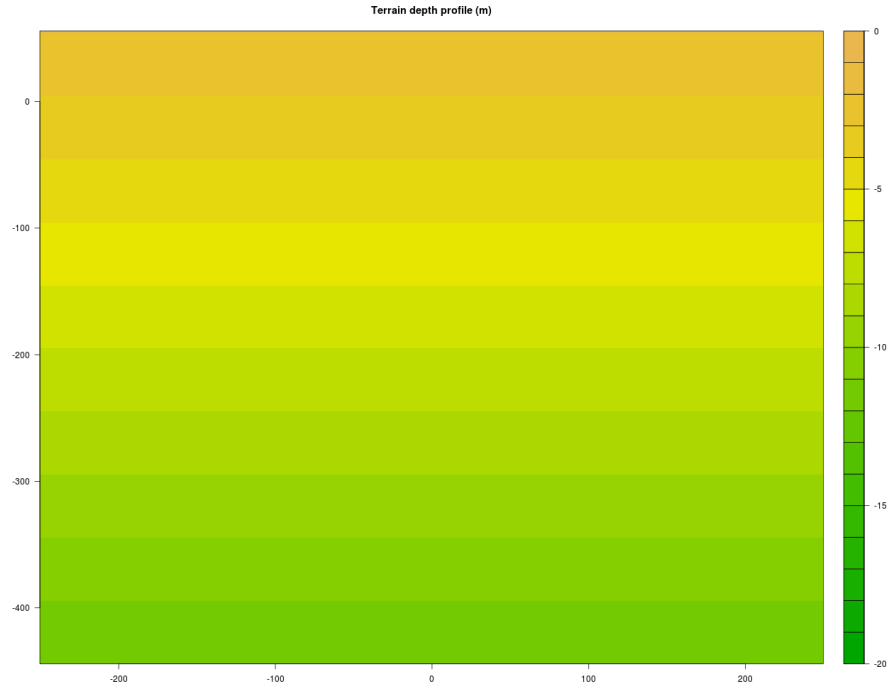
Algorithm 2: Ramer-Douglas-Peucker algorithm

```
1  function RamerDouglasPeucker (P, ε);
   input  : P = {p_0, p_1, ..., p_n}, epsilon
   output: P' = {p'_0, p'_1, ..., p'_m}, P' ⊆ P, m ≤ n
2  dmax = 0;
3  index = 0;
4  for i ← 2 to (length(P)-1 do
5  |  d ← PerpendicularDistance(P_i,Line(P_1,P_n);
6  |  if d > dmax then
7  |  |  index ← i;
8  |  |  dmax ← d;
9  |  end
10 end
11 if dmax ≥ epsilon then
12 |  P'_1 = RamerDouglasPeucker(P[p_1,...,p_index],epsilon);
13 |  P'_2 = RamerDouglasPeucker(P[p_index,...,p_n],epsilon);
14 |  P' ← P'_1 ∪ P'_2;
15 end
16 P = {p_0, p_1, ..., p_n};
17 return P';
```

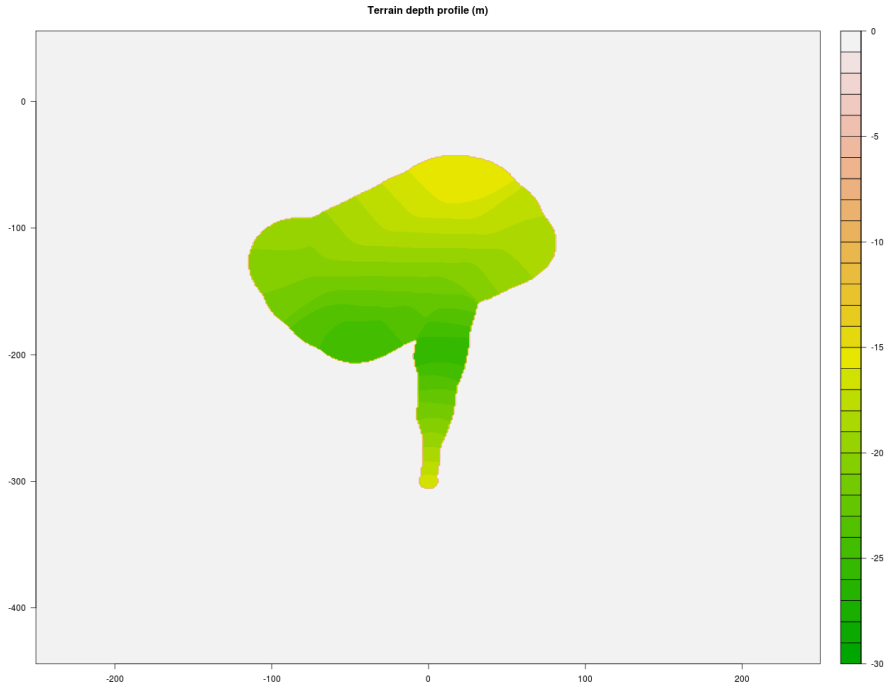
6.3.2 Map building

The information regarding the terrain topography needed for terrain representation is provided by a file, containing the DEM, consisting of terrain elevation for positions at regularly spaced horizontal intervals. The DEM's data is structured as a grid of cells. The cell length represents the accuracy of the terrain representation in 3D space and this value is defined in the DEM file. A DEM file is arranged as an ASCII grid file containing, in its header, the cell length, minimum and maximum x and y values of the grid in UTM. Then elevation values of the grid cells and related uncertainty are ordered in rows in the rest of the file. Each entry also has the relative position of the cell in meters to the origin of the map. Figure 6.2b shows an example 3D occupancy grid map constructed from SSS/altimeter range measurements obtained during experimental trials.

To address potential errors and inaccuracies in the map used for mission planning, close-proximity range measurements provided by the vehicle's sensors during mission execution need to be used. Now, a technique that integrates this data is required to incrementally construct and maintain a 3D map of the covered environment onboard the vehicle, considering sensor noise and localization errors. The method has to be computational efficient and capable of building elevation maps that are sufficiently accurate for our application scenario. Building globally consistent elevation maps is computationally hard within large-scale environments since it requires to maintain the whole map in memory. One also has to consider that the robot operating in this environment

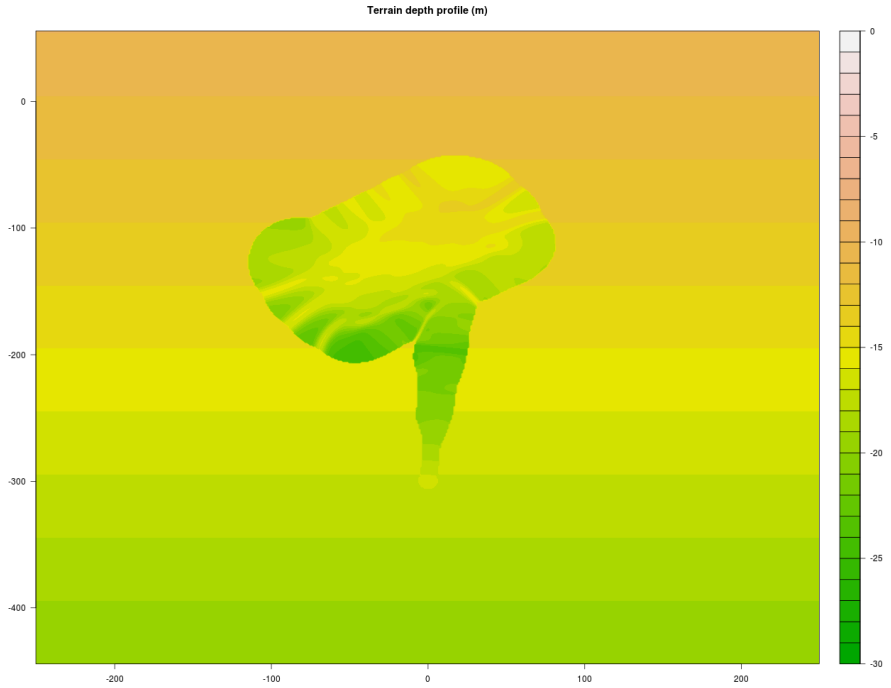


(a) Original DEM.



(b) Local DEM built from data acquired by a vehicle during a mission.

Figure 6.2: Graphical demonstration of the map building algorithm.



(c) Resulting DEM after combining a local map from a vehicle. The smoothness of the elevation measurements here is closely related to the variance of the elevation measurements on the original map.

Figure 6.2: Graphical demonstration of the map building algorithm.

has other task to execute and a mission to complete. Therefore it cannot spend all resources just for model updating.

Its is assumed that AUV's state measurements, including position and orientation, are given with known variance. The state measurements are sent by the INS system described in section 4.3. As the uncertainty of the localization measurements has a critical effect in the quality of the updated map, it is advisable to begin this procedure only after the vehicle has returned to surface. When the vehicle is at surface, our INS is updated with absolute positioning information through GPS fixes and the uncertainty in past measurements is reduced, as detailed in section 4.3.6.

Miller and Campbell (Miller and Campbell, 2007) propose a mixture-model based technique, where the elevation of each grid cell is modeled as a mixture of Gaussian elevation estimates. This algorithm provides an estimate of the elevation uncertainty in each grid cell. Miller's mixture model based technique was selected because it provides an elevation map represented probabilistically. Most techniques essentially insert local maps into a global map, but don't allow uncertainty associated with the map to be updated in the process. The local terrain maps are generated using Miller's mixture model based approach which is good for memory scaling, accuracy and considers measurement errors.

Measurements are associated to multiple locations in the elevation model using a Gaussian sum conditional density to account for uncertainty in the measured elevation and in the location

of the measurement. Some important features are treating sensor measurements with a statistical model (handling multiple sources of uncertainty) and allowing a quick update of the terrain map. A graphical visualization of the procedure is presented in figure 6.2.

The basic steps of the procedure are:

1. statistically represent each sensor measurement;
2. associate measurements with grid cells to which it is most likely to correspond;
3. fuse measurements assigned to each grid cell;
4. merge new and old grid maps.

6.3.2.1 Sensor measurement statistical representation

Each raw sensor measurement $\hat{r}^{ENU}$, in the ENU (or navigation) frame, is an elevation measurement built from the original raw sensor measurements $\hat{r}$ and the sensor orientation parameters $\hat{p}$. The function $f(p,r)$ transforms the raw sensor measurements in the sensor frame to the ENU or navigation frame.

$$r^{ENU} = \begin{bmatrix} E \\ N \\ U \end{bmatrix} = f(p,r) \tag{6.1}$$

The true variables of the variables are r and p, while n_r and n_p are the measurement errors.

$$\hat{r} = r + n_r \tag{6.2}$$

$$\hat{p} = p + n_p \tag{6.3}$$

By approximating the function $f(p,r)$ using a first-order Taylor series the following function is obtained:

$$r^{ENU} \approx f(\hat{p},\hat{r}) + \left.\frac{\partial f}{\partial p}\right|_{p=\hat{p},r=\hat{r}} n_p + \left.\frac{\partial f}{\partial r}\right|_{p=\hat{p},r=\hat{r}} n_r = f(\hat{p},\hat{r}) + J_p(\hat{p},\hat{r})n_p + J_r(\hat{p},\hat{r})n_r \tag{6.4}$$

where J_r and J_p are the Jacobians of $f(\hat{p},\hat{r})$. A posterior estimate of the height measurement can be determined by calculating the expected value of the previous equation.

$$E[r^{ENU}] \approx f(\hat{p},\hat{r}) = \hat{r}^{ENU} = \begin{bmatrix} \hat{r}^{EN} \\ \hat{u} \end{bmatrix} \tag{6.5}$$

The measurements uncertainty is given by the mean square error (MSE) matrix of the posterior measurement estimate.

$$P_{\hat{r}} = E[(r^{ENU} - \hat{r}^{ENU})(r^{ENU} - \hat{r}^{ENU})^T] \approx J_p(\hat{p},\hat{r}) Q_p J_p^T(\hat{p},\hat{r}) + J_r(\hat{p},\hat{r}) Q_r J_r^T(\hat{p},\hat{r}) = \begin{bmatrix} P_{\hat{r}}^{EN} & P_{\hat{r}}^{EN,u} \\ P_{\hat{r}}^{u,EN} & P_{\hat{r}}^{u} \end{bmatrix} \tag{6.6}$$

The Q_r and Q_p variables are the covariance matrixes of the $\hat{r}$ and $\hat{p}$ estimates. The previous equation outputs a matrix that describes the size and correlation of the georeferenced height measurement errors.

6.3.2.2 Measurement association to grid cells

Measurement correspondence to grid cells is uncertain due to the uncertainties both in the location where the measurement was taken and in the height measurement itself. The ENU joint probability distribution is given by $p_i(E,N,U)$, which is approximated as a multivariate Gaussian using the posterior measurement ($\hat{r}_i^{ENU}$ and covariance matrix P_i^{ENU}.

$$p_i(E,N,U) \approx \mathcal{N}(\hat{r}_i^{ENU}, P_i^{ENU}) \tag{6.7}$$

Applying the Gaussian transform, the EN joint probability distribution becomes:

$$p_i(E,N) \approx \mathcal{N}(\hat{r}_i^{EN}, P_i^{EN}) \tag{6.8}$$

The probability that a given terrain measurement $\hat{r}_i$ belongs to a specific cell j is described by:

$$P(\hat{r}_i^{EN} \in j) \int_{Ej^-}^{Ej^+} \int_{Nj^-}^{Nj^+} p_i(E,N)\, dN\, dE \approx \Delta E \Delta N p_i(E_j, N_j) \tag{6.9}$$

The integral is approximated as a single Riemann sum in order to allow its computation in real-time.

6.3.2.3 In-cell elevation distribution

Measurements must be assigned to grid cells according to the probability that they belong to those cells. Hence, it is required an univariate distribution of elevation measurements for each cell. A posterior univariate elevation estimate and related covariance of the measurement i is defined as:

$$\hat{U}_{i\in j} = \hat{u}_i + P_i^{u,EN}(P_i^{EN})^{-1}[EN_j - \hat{r}^{EN}] \tag{6.10}$$

$$\sigma^2_{\hat{U}_{i\in j}} = P_i^u - P_i^{u,EN}(P_i^{EN})^{-1} P_i^{EN,u} \tag{6.11}$$

As the a posteriori distributions of the elevation measurements are all Gaussian, the elevation distribution within cell j considering all measurements, $p(U_j\big|_{\hat{p},\hat{r}})$, is a Gaussian sum or Gaussian mixture calculated from these measurements:

$$p(U_j\big|_{\hat{p}_{1\ldots M},\hat{r}_{1\ldots M}}) = \frac{\sum_{i=1}^{M} p_{i\in j}\mathcal{N}(\hat{U}_{i\in j},\sigma^2_{\hat{U}_{i\in j}})}{\sum_{i=1}^{M} p_{i\in j}} \tag{6.12}$$

The previous model is not computationally feasible due to the fact that that are a number of measurements associated with each cell, affecting the size of the Gaussian mixture. Considering just the first and second moment of this Gaussian mixture, the j^{th} cell's elevation is characterized by the mean and covariance of the Gaussian mixture:

$$\hat{U}_{GM,j} = \frac{\sum_{i=1}^{M} p_{i\in j}(\hat{U}^2_{i\in j})}{\sum_{i=1}^{M} p_{i\in j}} \tag{6.13}$$

$$\sigma^2_{GM,j} = \frac{\sum_{i=1}^{M} p_{i\in j}(\hat{U}^2_{i\in j}+\sigma^2_{\hat{U}_{i\in j}})}{\sum_{i=1}^{M} p_{i\in j}} - \hat{U}^2_{GM,j} \tag{6.14}$$

6.3.2.4 Merging old and new maps

At this moment only a separate image of the DEM, built solely from the gathered elevation measurements, is available. The next step involves updating the local image of the DEM held by the robot for mission planning. The updating procedure should obviously improve the capability of the DEM to represent the terrain surface. So, the following technique is used for merging of old and new data, assuming that the elevation values to be fused have the same position. In the area to be updated, the merging can be performed by applying a weight function to the old and new data taking into account the uncertainty of the data. A satisfactory solution is the weighted mean value considering the variances of the two data sets:

$$u = \frac{\frac{1}{\sigma^2_o}z_o + \frac{1}{\sigma^2_n}z_n}{\frac{1}{\sigma^2_o}+\frac{1}{\sigma^2_n}} \tag{6.15}$$

$$\sigma = \frac{1}{\frac{1}{\sigma^2_o}+\frac{1}{\sigma^2_n}} \tag{6.16}$$

6.3.3 Clustering

After analysing mission performance by estimating the path that was followed by the vehicle and calculating the sensor's range to the cells in the updated grid, the next step is identifying areas that need better coverage:

- find cells with coverage below minimum;
- organize these cells in different groups;
- create areas that contain each group.

There are two approaches to hierarchical clustering: one goes "from the bottom up", grouping small clusters into larger ones, other goes "from the top down", splitting big clusters into smaller ones. These are called agglomerative and divisive clusterings, respectively. Here, agglomerative clustering was chosen as it suits our problem better, considering the need to agglomerate cells and the existence of a large number of identified areas. In agglomerative hierarchical clustering each object initially represents a cluster of its own. Then clusters are successively merged until the desired cluster structure is obtained.

Since clustering is the grouping of similar instances/objects, some sort of measure that can determine whether two objects are similar or dissimilar is required. In a Euclidean space it is possible to summarize a collection of points by their centroid, which is the average position of those points:

$$c_i = \frac{1}{n_i} \sum_{p \in C_i} p \tag{6.17}$$

Thus, if distance is considered as a measure of similarity then merging clusters whose centroids are at the shortest distance is a straightforward rule.

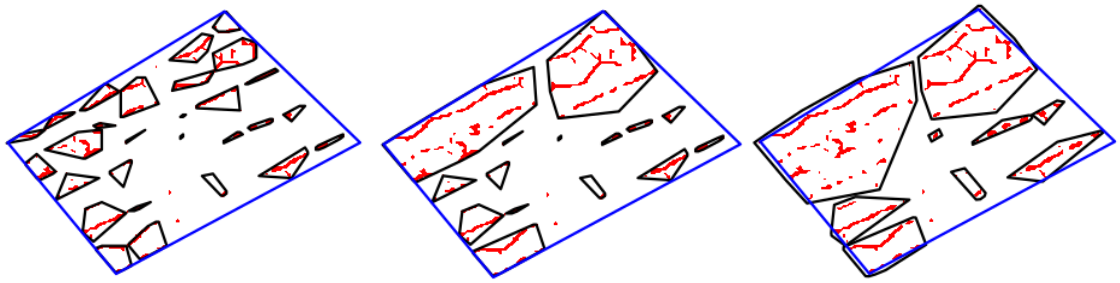
Single-link clustering defines the distance between two clusters as the minimum distance between their members:

$$d_{min}(C_i, C_j) = \min_{p_i \in C_i, p_j in C_j} \| p_i - p_j \| \tag{6.18}$$

It's called "single link" because it says clusters are close if they have even a single pair of close points, a single "link". This can handle quite complicated cluster shapes. This algorithm only wants separation, and doesn't care about compactness or balance. Single-link clustering has a drawback known as the "chaining effect" which happens when a few points that form a bridge between two clusters cause the single-link clustering to unify these two clusters into one.

Complete-link clustering consider the distance between two clusters to be equal to the longest distance from any member of one cluster to any member of the other cluster. This method usually produces more compact clusters.

$$d_{max}(C_i, C_j) = \max_{p_i \in C_i, p_j in C_j} \| p_i - p_j \| \tag{6.19}$$



(a) Phase 1. (b) Phase 2. (c) Phase 3.

Figure 6.3: Graphical demonstration of the clustering algorithm.

In order to solve our clustering problem, a personalized strategy to merge the clusters needs to be developed. In summary, our strategy consists of 3 phases that can be visualized in figure 6.3:

1. first clustering phase: merge clusters using centroid distance;
2. second clustering phase: merge clusters using single link similarity;
3. define areas to cover.

6.3.3.1 First clustering phase

Here, each cluster is represented by its centroid. The distance between clusters becomes the distance between its centroids. The algorithm searches for clusters close to each other and merges them, updating the corresponding centroids afterwards. Finding the best value for minimal cluster similarity, which is inversely proportional to the centroid distance, is problematic because if this value is too small then clusters will be big (the distances from points inside each cluster to its centroid are higher) and there will be a smaller amount of clusters. If this value is too high then many small clusters will be created (smaller distances between points and its centroid).

6.3.3.2 Second clustering phase

When adjacent clusters start getting too big they become harder to merge because the distance between its centroids is too big. This problem can be solved by adding a second clustering phase that merges clusters by measuring the minimal distance between them. Since the goal is to solve a coverage problem, the shape and size of the areas that will be created by the identified clusters needs to be analysed. The area is defined by the convex hull containing the identified cluster. Hence an area may contain cells that don't need better coverage, and doing that decreases the efficiency of the process. The $settings_{hullmaxratio}$ was introduced to establish a lower bound on efficiency of the coverage task. The ratio between the area of the hull around a new cluster and the area of the cells that need better coverage is compared to this parameter and the new cluster is only accepted if the ratio is inferior.

Algorithm 3: First Phase Clustering Algorithm.

```
1  function FirstClusteringPhase (grid, settings_minsimilarity);
   input  : grid(i, j), i = 1,...,n & j = 1,...,m; settings_minsimilarity
   output: clusters = {clusters_0, ..., clusters_p}
   /* Initialize clusters and centroids to individual cells      */
2  for i ← 1 to length(grid) do
3      for j ← 1 to length(grid_i) do
4          clusters.add(cell_ij);
5          centroids.add(GetCentroid(clusters.last));
6      end
7  end
   /* Initialize similarity matrix (triangular)                  */
8  for i ← 1 to length(clusters) do
9      for j ← i+1 to length(clusters) do
10         similaritymatrix_ij = ClusterSimilarityCentroidDistance(centroids_i, centroids_j);
11     end
12 end
13 while size(clusters) > 1 do
       /* Find most similar clusters                             */
14     for i ← 1 to length(similaritymatrix) do
15         for j ← i+1 to length(similaritymatrix_i) do
16             sim = similaritymatrix_ij;
17             c_1 = i;
18             c_2 = j;
19         end
20     end
       /* If clusters are not close enough then terminate and return
          clusters                                               */
21     if sim < settings_minsimilarity then
22         return clusters;
23     end
       /* merge most similar clusters                            */
24     clusters.add(Merge(clusters_c1,clusters_c2));
25     clusters.remove(c_1);
26     clusters.remove(c_2);
       /* update centroids                                       */
27     centroids.remove(c_1);
28     centroids.remove(c_2);
29     centroids.add(GetCentroid(clusters.last));
30     //* update similarity                                     */
31     similaritymatrix.removecolumn(c_1);
32     similaritymatrix.removecolumn(c_2);
33     similaritymatrix.removerow(c_1);
34     similaritymatrix.removerow(c_2);
35     for i ← 1 to length(similaritymatrix) do
36         similaritymatrix_i.add(ClusterSimilarityCentroidDistance(centroids_i,
           centroids.last));
37     end
38 end
39 return clusters;
```

Algorithm 4: Second Phase Clustering Algorithm.

```
1  function SecondClusteringPhase (grid, settings_minsimilarity);
   input  : clusters = {clusters_0, ..., clusters_p}; settings_minsimilarity; settings_hullmaxratio
   output: clusters = {clusters_0, ..., clusters_q}
   /* clusters and centroids are initialized to phase 1 output    */
   /* Initialize similarity matrix (triangular)                   */
2  for i ← 1 to length(clusters) do
3      for j ← i+1 to length(clusters) do
4          similaritymatrix_ij = ClusterSimilaritySingleLink(clusters_i, clusters_j);
5      end
6  end
7  while size(clusters) >1 do
       /* Find most similar clusters                              */
8      for i ← 1 to length(similaritymatrix) do
9          for j ← i+1 to length(similaritymatrix_i) do
10             sim = similaritymatrix_ij;
11             c_1 = i;
12             c_2 = j;
13         end
14     end
       /* If clusters are not close enough then terminate and return
          clusters                                                */
15     if sim < settings_minsimilarity then
16         return clusters;
17     end
       /* Test cluster candidate                                  */
18     clustercandidate = Merge(clusters_c1, clusters_c2);
19     hull = ConvexHull(clustercandidate);
       /* area occupied by the cluster                            */
20     clusteroccupiedarea = size(clustercandidate)*cellsize;
21     hulloccupiedarea = CalculatePolygonArea(hull);
22     oversizeratio = hulloccupiedarea/clusteroccupiedarea;
23     if oversizeratio < settings_hullmaxratio then
           /* accept new cluster                                  */
24         clusters.add(clustercandidate);
25         clusters.remove(c_1);
26         clusters.remove(c_2);
           /* update similarity                                   */
27         similaritymatrix.removecolumn(c_1);
28         similaritymatrix.removecolumn(c_2);
29         similaritymatrix.removerow(c_1);
30         similaritymatrix.removerow(c_2);
31         for i ← 1 to length(similaritymatrix) do
32             similaritymatrix_i.add(ClusterSimilaritySingleLink(clusters_i, clusters.last));
33         end
34     end
35 end
36 return clusters;
```

6.3.3.3 Area boundary definition

Finally, once the clusters are determined, the areas that will be used in the replanning process can be defined. As was mentioned earlier, the areas are convex hulls that contain each cluster. If the area is too small and if using a SSS (with nadir gap) it may not be possible to cover all the cells inside each area. For this reason the polygon that defines each area (away from its centroid) is enlarged, allowing coverage from outside the original area. The distance by which the area will be enlarged is calculated considering the average nadir width size:

$$nadirwidth = centroids_i.depth \times tan(settings_{sonar\_min\_angle}); \tag{6.20}$$

where $centroids_i.depth$ is the average depth of the cells in cluster i and $settings_{sonar\_min\_angle}$ is the minimal angle of the sidescan sonar.

6.3.3.4 Clustering evaluation

The quality of the created clusters is evaluated using the silhouette coefficient, which is a cluster validity measure.

$$cl_{sil}(i) = \frac{cl_{coh} - cl_{sep}}{\max(cl_{coh}, cl_{sep})}, -1 \leq cl_{sil} \leq 1 \tag{6.21}$$

The silhouette coefficient cl_{sil} combines two measures named cluster cohesion cl_{coh} and cluster separation cl_{sep}, both for individual points as well as clusters and clusterings. Cluster cohesion measures how closely related are objects in a cluster. It is determined by calculating the average distance between points inside a cluster. Cluster separation measures how distinct or well separated a cluster is from other clusters. It is determined by calculating the average distance between points inside different clusters. If a set of points can be clearly grouped into clusters, then it can be expected that the distance between clusters will be large compared to the radius of the clusters. This is the reason why the silhouette coefficient is an important measure. Calculating the average cluster cohesion and separation enables us to calculate the average silhouette coefficient of the clustering. This is a measure of how tightly grouped all the points are. Negative values indicate that the cluster radius is greater then the distance between clusters, so that clusters overlap, suggesting poor clustering performance. Large values suggest good clustering. These measures of clustering density are important to mission planning because, for example, it can help to determine if reusing previously calculated mission plans may be more efficient then try to cover multiple small areas with sparsely distributed points that would need additional coverage.

6.3.4 CBR replanning

Consider the following definitions:

$$A = \{a_1, a_2, ..., a_n\} \tag{6.22}$$

$$S^i = \{s_1, s_2, ..., s_{n_i}\}, i = 1, ..., n \tag{6.23}$$

$$A' = \{a'_1, a'_2, ..., a'_m\}, A' \in A \tag{6.24}$$

$$S'^j = \{s'_1, s'_2, ..., s'_{n_j}\}, j = 1, ..., m \tag{6.25}$$

where S^i denotes the set of solutions found in the search area a_i by the evolutionary offline planner. The new areas A', determined previously by the clustering algorithm, are a subset of the original areas A used for offline planning. In the worst case scenario where the true mission performance is below the requirements in all of the coverage area, A' may become equal to A. The online coverage problem is then the search problem of finding solutions S'^j to the new areas A', optimizing the objective function's value considering previous search performance.

It is assumed that the online planner will experience similar problems over time. If this is the case, then one can benefit from having previously solved those problems and reuse the identified solutions. Hence, the CBR methodology is adopted and S^i, the Pareto sets found for each area a_i, is considered our casebase. Each dominant individual found by the evolutionary planner can be interpreted as a case since they already hold all relevant information about the solution they describe.

When confronted with a new problem, the CBR module will determine the level of similarity between this problem and the one solved in offline planning. In order to assess this similarity, a similarity metric is needed. CBR research has shown that defining a problem's similarity metric is non-trivial. Problem similarity measures tend to be domain (or problem) specific so a similarity metric specifically tailored to our application domain will be devised. The main challenge here is dealing with partially unknown environments since the offline planning algorithm optimizes the solutions to the available image of the environment at planning time, which my be inexact. For this reason a statistical measure of error E_{map} was adopted, which describes the difference between original and updated environment maps, as our measure of similarity.

The system architecture is presented in Fig. 6.4. Given a new task to cover the identified areas, it can choose between planning using the previous path, locally optimizing a path from the casebase or even search for a new one. Case-based path planning and evolutionary path planning are complimentary methods to achieve better mission performance in dynamic environments or in scenarios where only imperfect information is available. The evolutionary path planner seeds the casebase with innovative, dominant and diverse solutions. The case-based path planner remembers the characteristics of the solutions determined in the past and uses them to solve new problems similar to the original. It is up to the online planner to decide whether to use an old well-tried solution from a casebase or to look for a completely new alternative solution to the problem. Once the solution is found, it will be sent for execution by the planner.

Next, a description of the replanning system is provided considering the CBR framework.

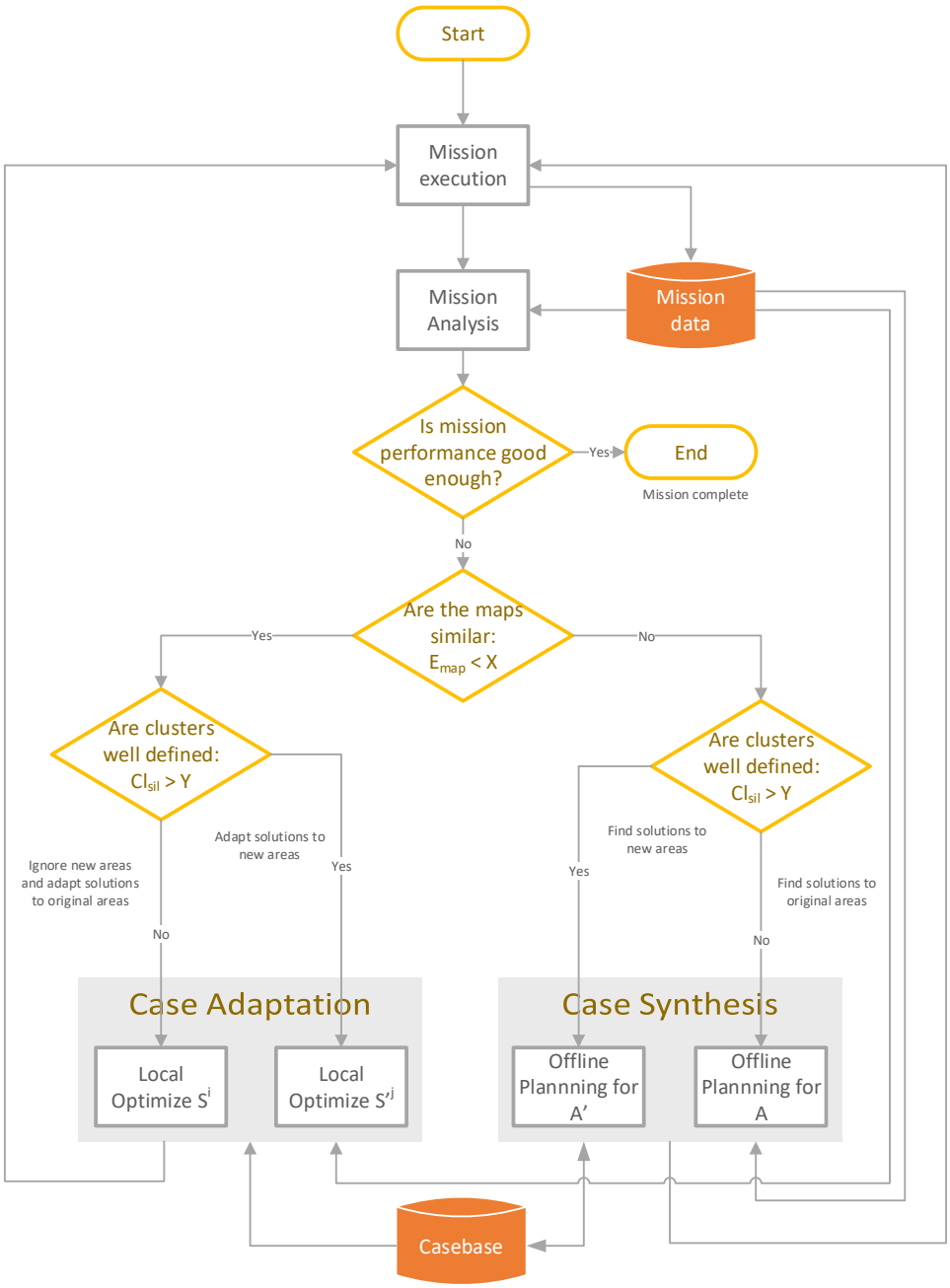


Figure 6.4: Mission replanning flowchart.

6.3.4.1 Case Retrieval

Before deciding which parameters will be used for generating area solutions for the current problem, we need to reevaluate the performance of the cases in the casebase according to the new information. The procedure is:

- using the estimated trajectory that was followed by the vehicle (see section 6.3.1) and the updated map image (see section 6.3.2), estimate the detection performance;
- estimate or measure the energy spent by the vehicle;
- select the dominant cases for further consideration.

6.3.4.2 Case Adaptation

Unless no area was previously covered, then there is always some sort of adaptation performed to the old paths. This task involves the following steps:

- for each area a'_i, identify the original area a_i that contains it;
- using the path parameters from selected cases in the casebase for original area a_i, generate solutions for each new area a'_i; each area a'_i now has a set of candidate paths; paths belonging to distinct areas a'_i that are contained in the same original area a_i share the same parameters;
- perform local optimization of every path of every area a'_i (see section 5.4), optimizing the path to the new area and/or the updated map;
- update detection performance, energy consumption and mission time duration estimates;

6.3.4.3 Case synthesis

New paths need to be synthesized when no useful or adaptable paths are found in the casebase. This may happen if the updated map is substantially different then the original used to build the casebase. Essentially, it is a new problem. When more extensive search is needed, the offline planner is used. At every generation during the execution of the EA for online path planning, the EA refers to the current world model held by the vehicle in order to evaluate the fitness of each individual solution in a population. The adaptivity of the EA will cause modifications in the current individuals in response to changes of the world model due to input of data from sensors. Typically, a EA randomly initializes its starting population so that it can get an unbiased sample of the search space. However, since problems do not usually exist in isolation and, in our case, there is always some information about the environment (although it may be incomplete) the problems will be similar to some extent. In this case it makes little sense to start a search attempt from scratch when previous searches may have yielded useful information about the problem domain. Instead, the online EA starts with a population identical to the population that the offline EA had lastly. If none of the previous individuals are useful for solving the new problem, they are removed from

the population and new randomly initialized individuals are generated as the starting population for the search.

In the same belief that similar problems could have similar solutions (Leake and Wilson, 1999), it may be worthwhile using networks that have similar synaptic weights to deal with similarly structured environments, or at least to initialize newly created network synaptic weights from the point where the old trained one has stopped, instead of initializing it randomly. If the training error is too high then start training from scratch considering the reevaluated individuals. In other words, generate a solution on the basis of already acquired knowledge and experience in the memory.

6.3.4.4 Casebase Management

Remembering past experiences is a essential requirement to implement the learning system. However, when problems scale up, forgetting becomes as important as remembering. To keep the size of the casebase constrained and to have a better overview of the learning process, a new case is stored only if there is no similar case in a sense. The old experiences can be replaced with the similar but better ones in terms of fitness, resulting in the casebase containing only a few examples from each cluster of similar solutions. Storing only distinct cases slows down memory consumption but does not guarantee that the casebase will stay constrained.

Details about our strategy are explained in section 5.7. Essentially, the density of individuals in decision space is controlled for each casebase. The complete Pareto set (dominant solutions) is kept in the casebase (the good and bad solutions, depending on which objective is being considered). Remembering cases with different performance on each objective can play a positive role in the decision making process. A worse local solution may need to be chosen if the global planner cannot find a better feasible solution.

6.3.4.5 Global Planner

After obtaining multiple solution candidates for each area a'_i, the search for a global solution needs to be performed, determining the order in which each area is visited, using a global planner. The algorithm for online global planning is the same iterative algorithm used for offline global planning 5.8.

6.3.4.6 Execution Time Constraints

Guarantee that algorithm is executed while the vehicle interrupts the mission to perform a surface manoeuvre in order to reduce navigation error or communicate with the base station. Faster replanning time may be achieved by reducing the amount of solutions in the casebase but that may have an impact on the quality of the online solution.

6.4 Experiments

6.4.1 Parameter tuning

There are several alternative strategies to replan a mission and, at this point, it is not clear to us which is the best:

- optimize solutions to original areas;
- optimize solutions to new areas;
- offline planning on new areas initialized with best solutions found previously on original areas;
- offline planning on original areas initialized with best solutions found previously on original areas;
- offline planning with random initialization.

All strategies are capable of generating a solution but may greatly differ in performance and planning time. Our hypothesis is that each one of these strategies will suit a specific scenario. So the objective here is to clearly identify these scenarios in order to maximize the performance of the replanning algorithm under any conditions.

Each scenario defines an outcome of a mission that was executed previously. The scenario can be characterized by these features:

- measure of error E_{map}, describing the difference between original and updated environment map;
- number of identified clusters;
- cluster silhouette coefficient;
- cluster cohesion;
- cluster separation;
- percentage of area left uncovered.

Thus, it is important to understand which of these features will help determine the correct replanning strategy. It may be necessary to identify a range of values for a feature (choosing a set of thresholds). For example, if the measure of map error E_{map} is higher then a specified threshold, then offline planning with random initialization may be the best option. On the contrary, if the map wasn't updated (the map used for planning was accurate, close to the real map) and there is a clearly identified area that wasn't covered then simply adapting the previous solution to the new area may be good enough. Choosing the proper values for these thresholds is a complex task because the relationship between the scenarios and the replanning strategies is not known in

advance. Hence, it is essential to define a test procedure that evaluates the relevant measures (and their thresholds) by comparing the replanning algorithm performance under different scenarios.

To compare the performance of different replanning strategies on a specific scenario, the following measures of performance are determined:

- increment in POD;
- amount of uncovered area;
- amount of resources used;
- replanning time.

Algorithm 5: Algorithm for testing replanning techniques

```
1  function TestReplanningTechniques output: Scenarios, PerformanceData
   /* Execute a previously planned Mission                              */
2  MissionData = ExecuteMission();
   /* Test distinct E_map                                               */
3  for i ← 1 to M do
       /* Changing variance from map measurements will change the
          updated map                                                   */
4      v = ChangeVariance();
5      UpdatedMap = UpdateMap(MissionData,TopographicMap(v));
6      CalculatePerformance(OriginalArea,MissionData,UpdatedMap);
       /* Identify all clusters                                         */
7      ObtainAllClusters();
       /* Create cluster subsets with different characteristics         */
8      GetClusterSubsets();
9      for j ← 1 to length(ClusterSubsets) do
10         NewAreas = CreateAreasFromClusterSubset(ClusterSubsets[j]);
11         (Scenario[i+j], PerformanceData[i+j]) =
            TestReplanningTechniques(NewAreas,OriginalAreas,UpdatedMap);
12     end
13 end
```

The procedure for obtaining and testing different scenarios is described in algorithm 5. Our intention is to gradually increase the difference between the original map (used for offline planning) and the updated map (after performing a simulated mission) to assess the ability of the online planing procedures to generate solutions close to the ones given to the offline planners. It is expected that solutions will be further away as E_{map} increases because the online planner can't perform an exhaustive search like the offline counterpart. Essentially, an updated map is obtained by using measurements to update an initial version of the map. The map building process combines different representations of the map by applying a weight function to the old and new data taking into account the uncertainty of the data. This way it was possible to obtain different map error values by varying the variance of the data on the initial map (which was easier to do). It was also desired

to test sets of clusters displaying distinct separation and cohesion statistics. This was achieved by considering sets with distinct (uniformly distributed if possible) silhouette coefficients. This naturally affects the number of clusters used in each scenario and the percentage of area left uncovered. It is believed that the disposition of the clusters will help dictate which type of technique may be adequate. Each version of the updated map and subset of clusters represents a scenario and will be used by all replanning techniques for comparison. Almost 300 replanning simulations were performed during this procedure, taking several days to accomplish.

Having performed all these simulations, the resulting information needs to be analysed so a suitable methodology can be derived, enabling the correct replanning technique to be chosen according to the features of a given scenario. Given the amount of variables and thresholds to test, an evolutionary algorithm was implemented to find the most promising set of thresholds. Here, the individual is defined by a set of thresholds and undergo the actions of recombination and mutation. The best planning methodology, from the ones listed above, will always be the offline planning with random initialization since it performs an exhaustive search through all decision space, but it also takes a very long amount of time to do so. The other offline techniques are variations that will very likely speed up the planning procedure. Hence, the quality of the solution found by the other techniques was evaluated in terms of normalized distance in objective space to the solution given by the best offline procedure. An online planning procedure should provide a solution which is close to the best, but should do so under a reasonable amount of time. So, a minimum distance parameter was introduced to give priority to an online technique, otherwise the offline technique would always be chosen.

A detailed analysis of the data from the tests as well as the data from the evolutionary algorithm was performed and it was found that the most relevant features from a scenario are the map error statistic and the clustering silhouette coefficient. In figure 6.5 it can be seen that, as expected, as the map error increases the online replanning procedure becomes unable to produce solutions as good as the one produced by the offline planning procedure. Clearly testing a greater number of topographic maps would be beneficial, but that was not possible due to the amount of time taken by the simulations. In figure 6.6, it can be seen that as the clustering silhouette coefficient increases the quality of the solutions given by the planner that considers new areas (derived from clusters) increases. This is very important as it is not known in advance if there is an advantage in planning using new areas (not covered previously) or covering the original area all over again, and evaluating this may be very costly. A higher silhouette coefficient indicates better defined clusters while a lower silhouette coefficient indicates small and scattered clusters. Having many clusters may be bad as the vehicle spends too many resources travelling between them. From the figure it is possible to identify a clear trend and this allows us to define a minimum threshold on the silhouette coefficient that indicates a region where choosing the new areas may be advantageous. Making this decision is important, because as depicted in figure 6.7, the replanning time is proportional to the number of clusters if considering this approach. The other features did not allow us to conclude anything relevant. The evolutionary algorithm told us the same thing as the best individuals had arbitrarily fixed values on these other features. It was possible to identify the correct strategy with

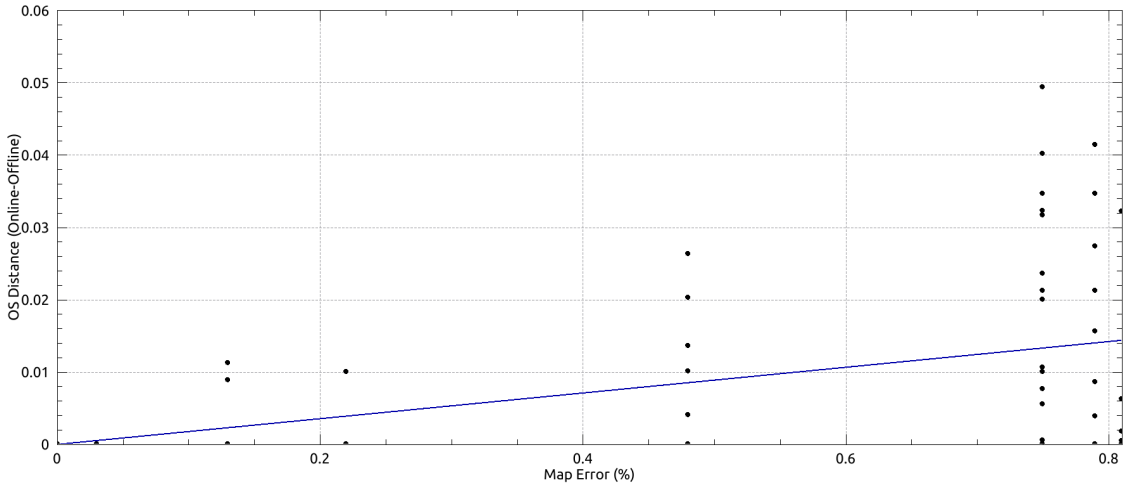


Figure 6.5: Plot relating distance in objective space between the best online and offline solutions to the map error statistic.

an accuracy of 80%, which is acceptable given that all replanning methods were able to produce valid solutions. In case the solutions are invalid, which occurs when there are not enough resources to complete the planned mission, then an alternative is to simply test other replanning method and take the replanning time penalty as a consequence.

6.4.2 Scenario

The Leixões harbor was chosen to demonstrate the capabilities of our planning methodology since it is a typical application scenario and it is our preferred testing location as it is close to our research institute. Leixões is one of Portugal's major seaports, located 4 km north from where the Douro River and the Atlantic Ocean converge, in Matosinhos municipality, near the city of Porto. Geospatial data describing the seafloor topography was obtained from a freely accessible database from the Portuguese Hydrographic Institute (www.hidrografico.pt). The dataset contains a grid with 100 meters resolution constructed with depths from the latest hydrographic surveys made on the area. Figure 6.8 shoes an aerial view from the harbor. The operation area is represented in white and its dimensions are approximately 100 by 80 meters or 8000 squared meters.

6.4.3 Simulations

Three simulation test cases demonstrate the applicability of the proposed planning methodology to typical coverage problems. In particular, the goal is to assess the influence that the navigation system's performance and the knowledge of the environment have on the search mission and, as a consequence, on the replanning stage. The simulations take place on the Leixões harbor, presented earlier, and involve the following test cases:

- test case A: low map error + bad navigation performance;
- test case B: low map error + average navigation performance;

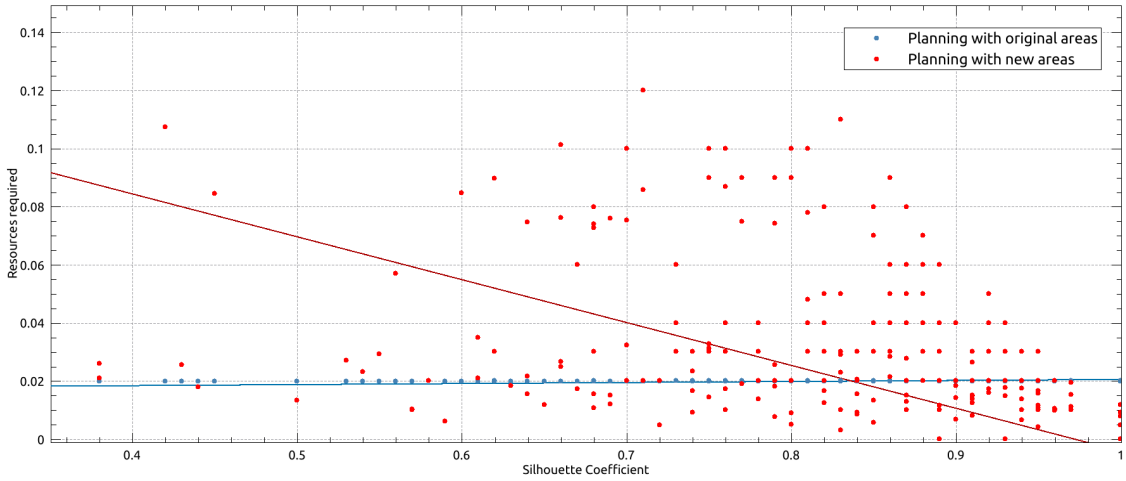


Figure 6.6: Plot relating the amount of resources required for a mission and the clustering's silhouette coefficient.

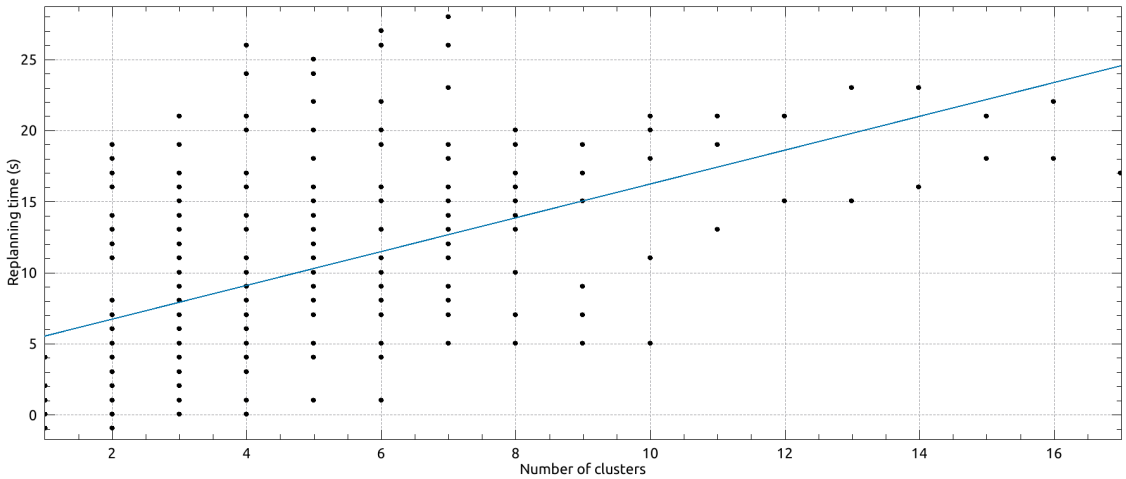


Figure 6.7: Plot relating the time required to replan a mission and the number of identified clusters.

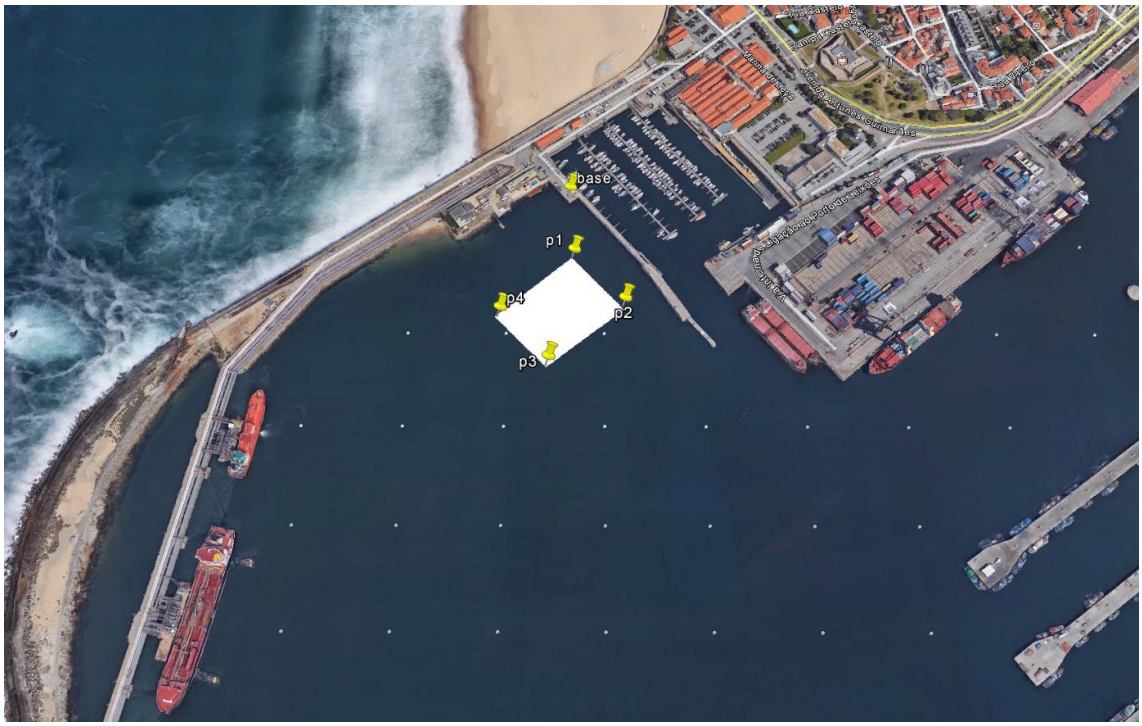


Figure 6.8: Google Earth satellite view of the Leixões harbor in Portugal. The operation area is represented by a polygon in white. The blue dots are the measurements from the geospatial database.

- test case C: high map error + good navigation performance.

Overall, these three distinct situations highlight the suitability and versatility of the presented planning methodology with respect to different mission outcomes. The complexity of the presented scenarios reveals the wide range of use-cases where the planner can be employed. For simulation purposes, a refined topographic grid was calculated, with size 500x500 and 1x1 squared meters cells, using the samples from the geospatial database presented earlier. This is the elevation grid used exclusively by the simulator to simulate the real world and allow realistic range measurements. The planner itself will first use a simplified planar grid to perform offline planning and later on, using the data acquired during mission execution, it will estimate its own updated image of the topographic grid. The depth and slope of the planar grid used by the planner will depend on the the degree of approximation of the grid used in each test case. AUV position was estimated using our INS, presented on section 4.3, which fused heading from the compass, absolute position from the GPS, depth estimated from pressure readings and velocity over ground measurements from the DVL. Orientation, position, velocity and acceleration are calculated by our simulator in each simulation step using the kinematic model for the MARES AUV, adding noise and bias before sending the data to control, navigation and planning systems. The performance for each test case is established by controlling the amount and quality of data sent by the simulator to the navigation system. Bad navigation performance is achieved by sending acceleration and angular velocity measurements with high noise and a drifting bias to the INS, without a velocity fix.

Average navigation performance is achieved by sending acceleration and angular velocity measurements with high noise and a drifting bias to the INS, with a velocity fix every 40 seconds. Good navigation performance is achieved by sending acceleration and angular velocity measurements with low noise (but without drifting bias) to the INS, with a velocity fix every 20 seconds. In all test cases, the INS is able to use GPS measurements to reduce uncertainty although such data was not actively requested. Since the vehicle is navigating in shallow water and the missions are relatively short, it receives GPS data at the start of the mission, at the end and occasionally during the mission if it is unable to precisely control the depth of the vehicle. Contributing to this instability is the obstacle avoidance system, presented in section 4.4, that is run by the simulator using a ray-tracing algorithm. This system actively controlled the altitude of the vehicle using the *flyover* behaviour exclusively, as it is unusual to find obstacles underwater or rough terrain in such a scenario.

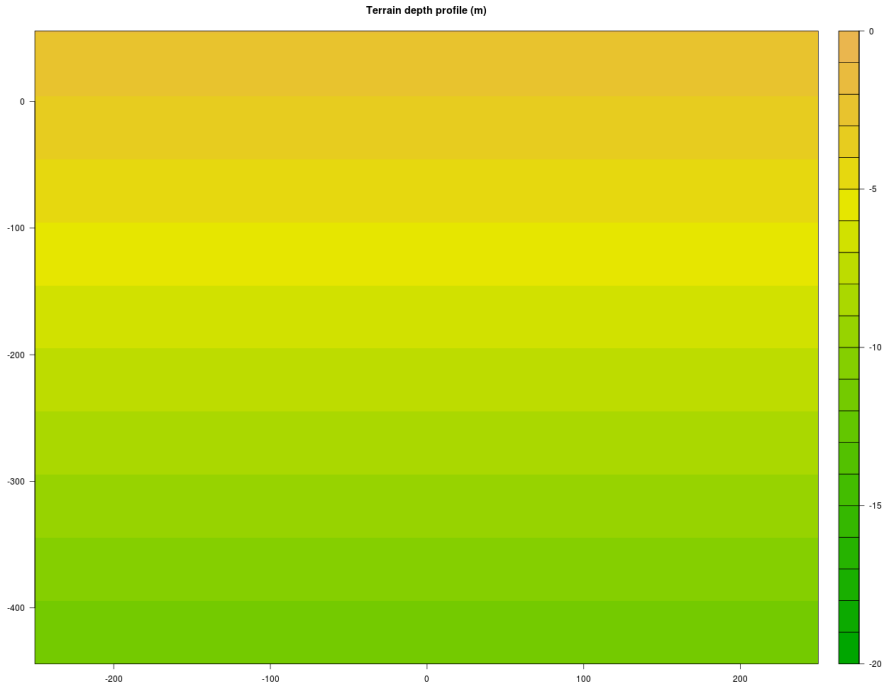
6.4.3.1 Offline planning

The offline planner executed the evolutionary algorithm, with terrain represented in figure 6.9b, for 109 seconds terminating on the fifteenth iteration. As displayed in figure 6.10a, a total of 12 solutions were identified by the planner and the chosen solution was the orange one at the top. It was the only solution that guaranteed full area coverage, although at the cost of higher energy consumption due to the higher vehicle velocity chosen (1.4 m/s). Figure 6.10b presents the chosen solution. Since the slope was relatively small, at around 1.15 degrees, all solutions were defined at constant depth and this is the main reason why the direction of the chosen solution is not aligned with the direction of the slope. It is important to keep all the other non dominant solutions, even if they have inferior detection performance, because during the mission, progress might be slower then expected due to environmental conditions or excessive need to perform surfacing manoeuvres and it might be necessary to revert to one of the other mission plans. They might not cover all situations but form a good starting point when there is a need to replan during the mission.

6.4.3.2 Test case A

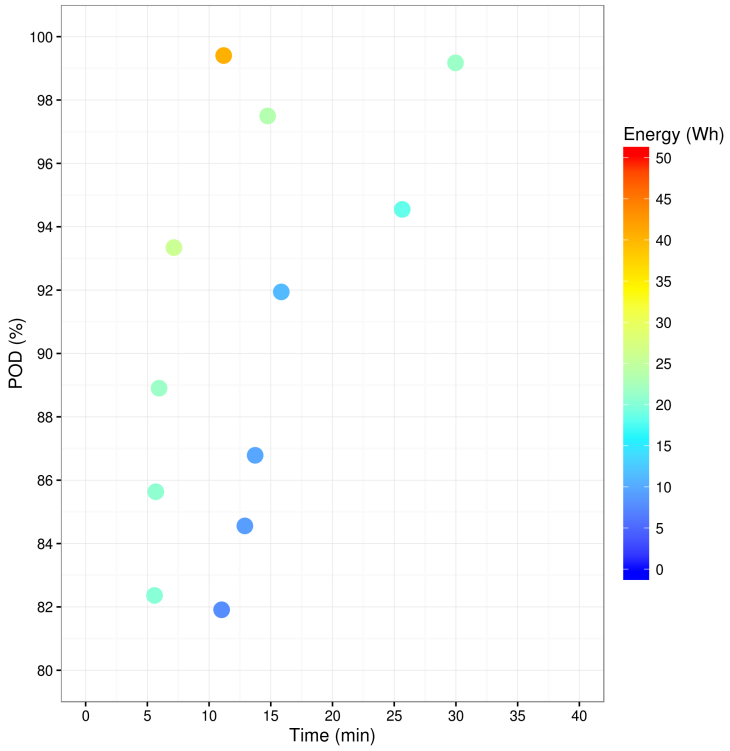
The planned and real paths followed by the vehicle during test A can be observed in figure 6.11. As expected, the INS performance without velocity fix was pretty bad and the vehicle was unable to follow the planned trajectory. With a silhouette coefficient of 0.86, the clusters were well defined, as can be seen in figure 6.12a. The map building process identified a 1.75% difference between the original map and the map built with acquired data during the mission. This lead the online replanner to choose as strategy to optimize previous solutions to the new operating areas, displayed in figure 6.12a in black. Replanning took 39 seconds and produced a solution that promised full area coverage with 99.8% probability of detection. The updated terrain and the new solution can be visualized in figure 6.12b. That divergence at the end (top left of the figure 6.11) was caused by an unrequested surface manoeuvre where the INS was able to acquire a GPS fix and as a result corrected the trajectory according to the current state of the mission.

(a) Terrain used for simulation. Uses real data from geospatial database.

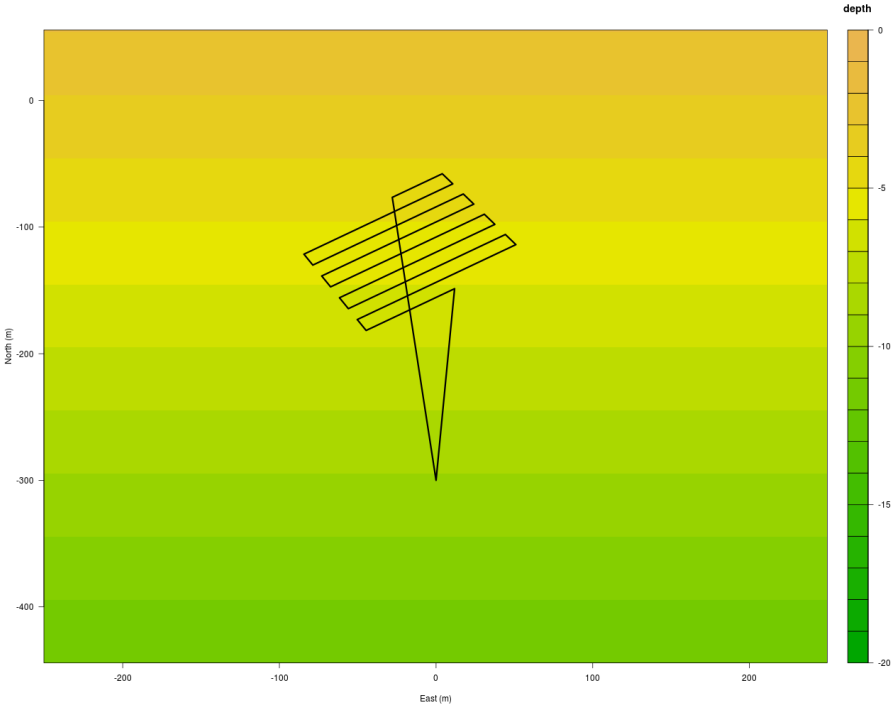


(b) Sloped terrain used for offline planning.

Figure 6.9: Terrains used for representing the topography in the Leixões harbor.



(a) Range of solutions identified by the evolutionary planner.



(b) Solution chosen by offline planning.

Figure 6.10: Solutions generated by the evolutionary planner.

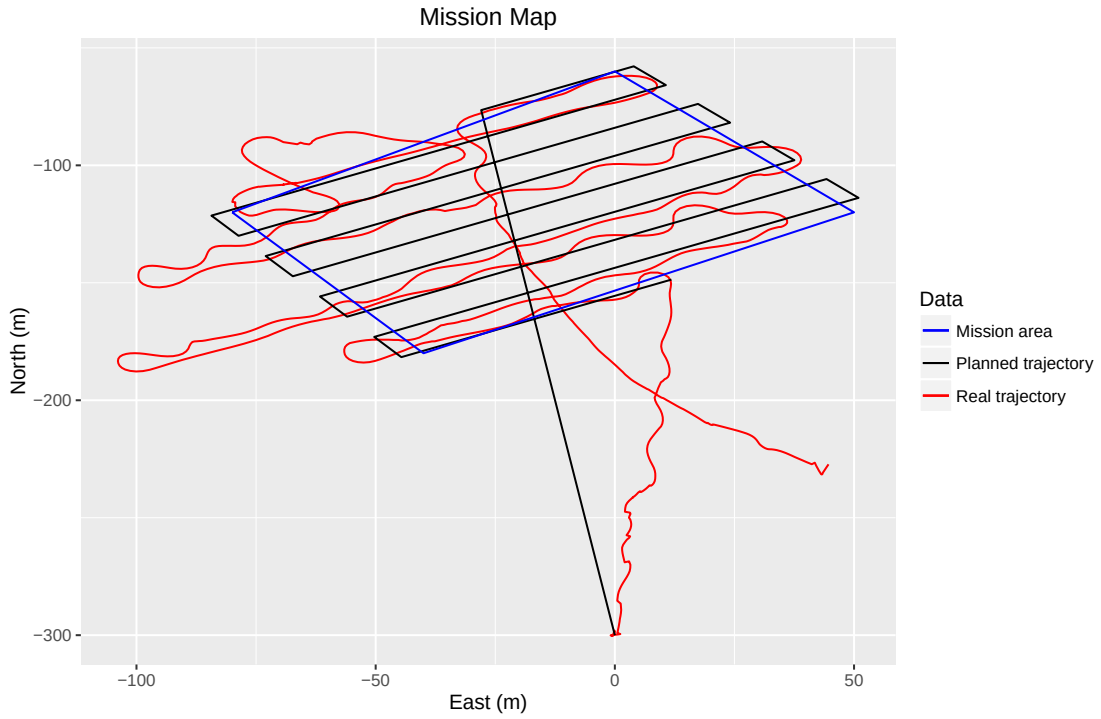


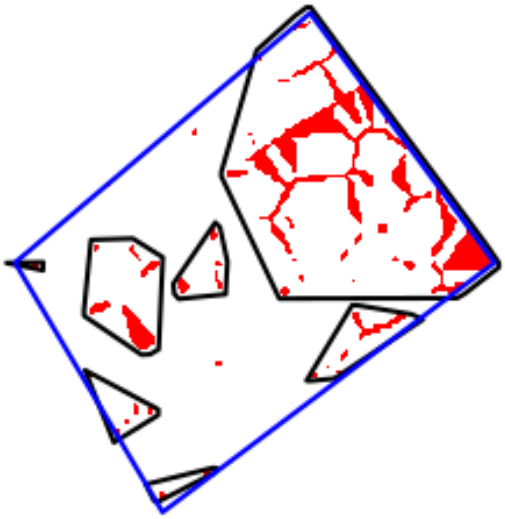
Figure 6.11: Plot of the trajectory generated by the offline planner and the real trajectory followed by the vehicle, for test case A.

6.4.3.3 Test case B

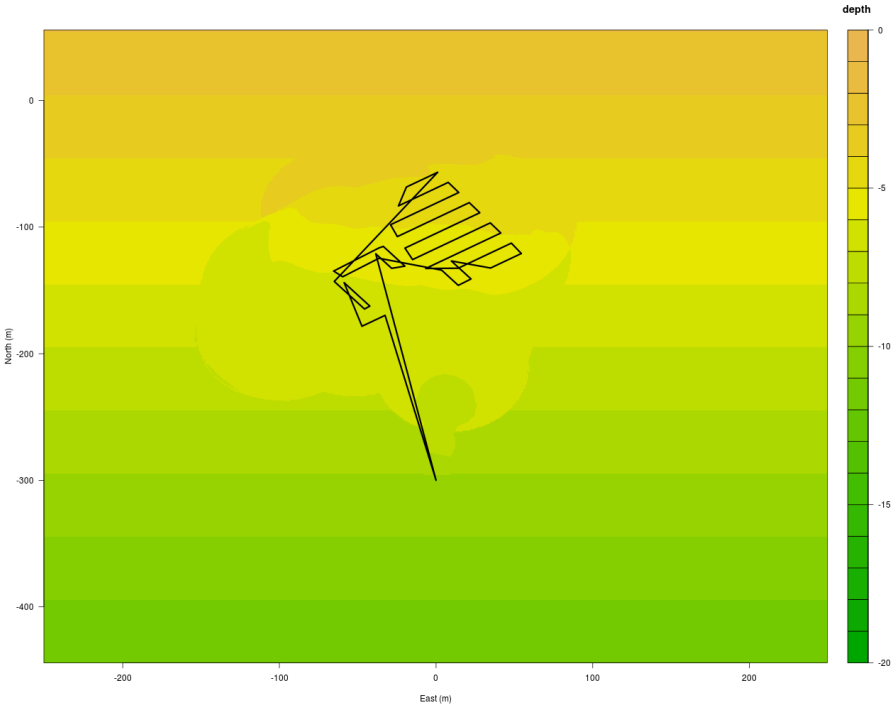
The planned and real paths followed by the vehicle during test B can be observed in figure 6.13. The INS performance with a velocity fix, even if sparse with one fix every 40 seconds, was better then expected and the vehicle was able to perfectly follow the planned trajectory. With a silhouette coefficient of 0.80, the clusters were scattered over the operation area, as can be seen in figure 6.14a. The map building process identified a 6.34% difference between the original map and the map built with acquired data during the mission. The higher confidence on the geolocalization of the samples led to this increase on the amount of update done to the map, when comparing with test case A. This lead the online replanner to choose as strategy to optimize previously found solutions to the original operating area, displayed in figure 6.14a in blue. Replanning took 28 seconds and produced a solution with 99.4% probability of detection of any object in the area. The updated terrain and the new solution can be visualized in figure 6.14b.

6.4.3.4 Test case C

According to the purpose of this test, a terrain in the form of a valley was used together with the simulator, replacing the terrain calculated from the real geospatial data. The effect of navigating in an partially unknown terrain with incorrect information can be better evaluated by maintaining the terrain used for offline planning. The planned and real paths followed by the vehicle during test C can be observed in figure 6.15. For our surprise, the performance of the INS, even with



(a) Identified clusters, showing area that was not covered with desired level of performance.



(b) Planned trajectory using online algorithm and updated map.

Figure 6.12: Output of the online planner for test case A.

higher fix rate, was equal to the one achieved on test case B. This confirms the capability of the lawnmower pattern of cancelling error growth in the presence of noise and bias drift on IMU data, as was the case on the previous test. The inability of the vehicle to follow the exact trajectory after turning is due to the performance of the controllers during acceleration. It can clearly be seen that once velocity stabilizes, the trajectory deviation is reduced. Although with a silhouette coefficient

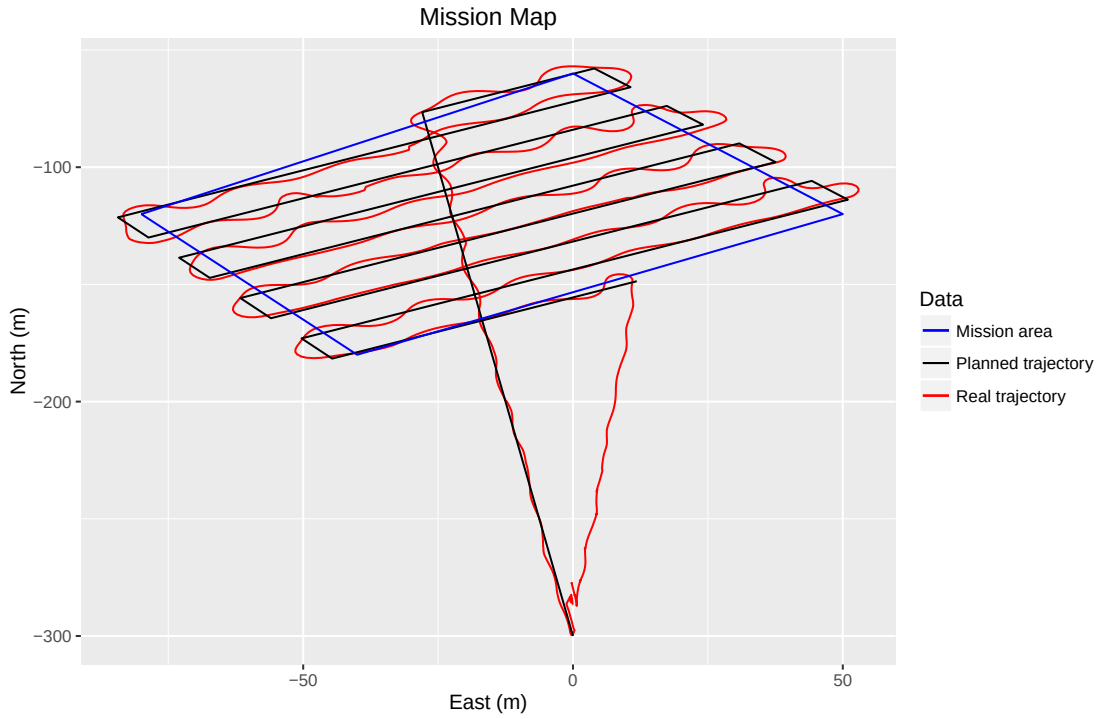


Figure 6.13: Plot of the trajectory generated by the offline planner and the real trajectory followed by the vehicle, for test case B.

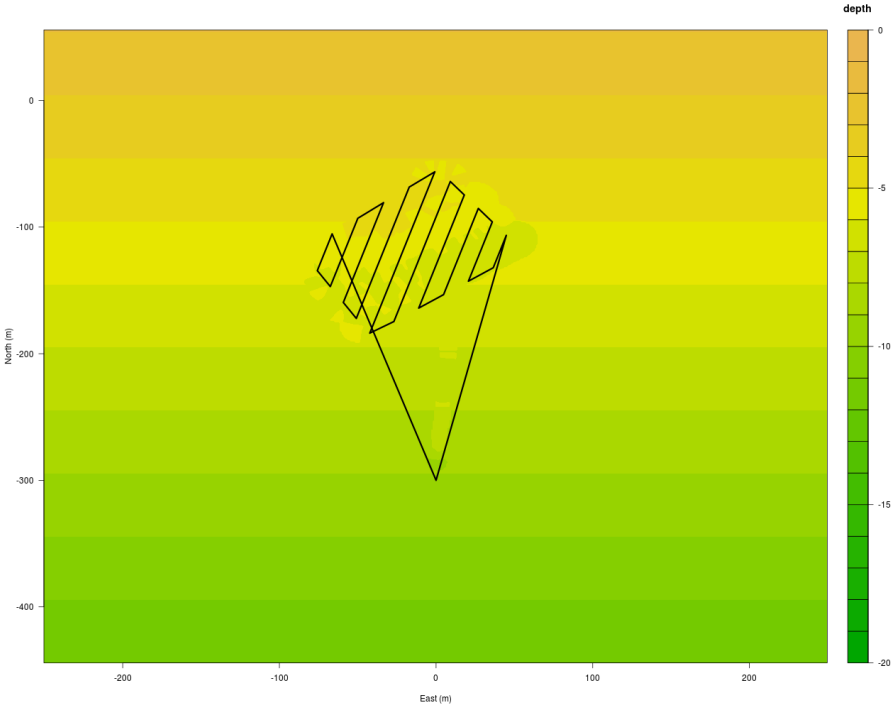
of 1.00, the identified cluster actually covered all of the the operation area, as can be seen in figure 6.16a. The map building process identified a 43.88% difference between the original map and the map built with acquired data during the mission. This lead the online replanner to choose as strategy to optimize previously found solutions to the new operating area, displayed in figure 6.16a in black, which coincidentally equal to the original area. Unfortunately, the chosen strategy was unable to generate a good solution to the problem and it was required to use the evolutionary planner. It needed 79 seconds to come up with a solution that guaranteed full area coverage with maximum detection performance. The updated terrain and the new solution can be visualized in figure 6.16b.

Following these successful simulations, the applicability of our planning methodology to real world search missions will be further evaluated in a real test presented in the next section.

6.4.4 Field trial

The final trial was also performed at the Leixões harbor. It was decided to use exactly the same operating area that was considered in the simulations in order to compare the performances of the vehicle and the simulator. Here, the best image of the terrain had to be used for preplanning (represented in figure 6.9a) calculated using the data from the geospatial database. The offline planner took 2 minutes to converge to the best set of solutions. The Pareto Front is represented in figure 6.17. It was found that the algorithm created many invalid solutions in the search process, which slowed the search. This reason for this is that the depth of the terrain inside the operating

(a) Identified clusters, showing area that was not covered with desired level of performance.



(b) Planned trajectory using online algorithm and updated map.

Figure 6.14: Output of the online planner for test case B.

area was quite low, so the EA could not create diverse enough solutions. The solutions were invalid because the constraints were not respected, in particular, the minimum operating depth and the minimum altitude the vehicle is allowed to maintain. The chosen solution provided 98% POD over the area and requires an estimated 30 minutes and 42 seconds to complete the mission.

After deploying the vehicle, it displayed an erratic behaviour as it didn't seem to follow the

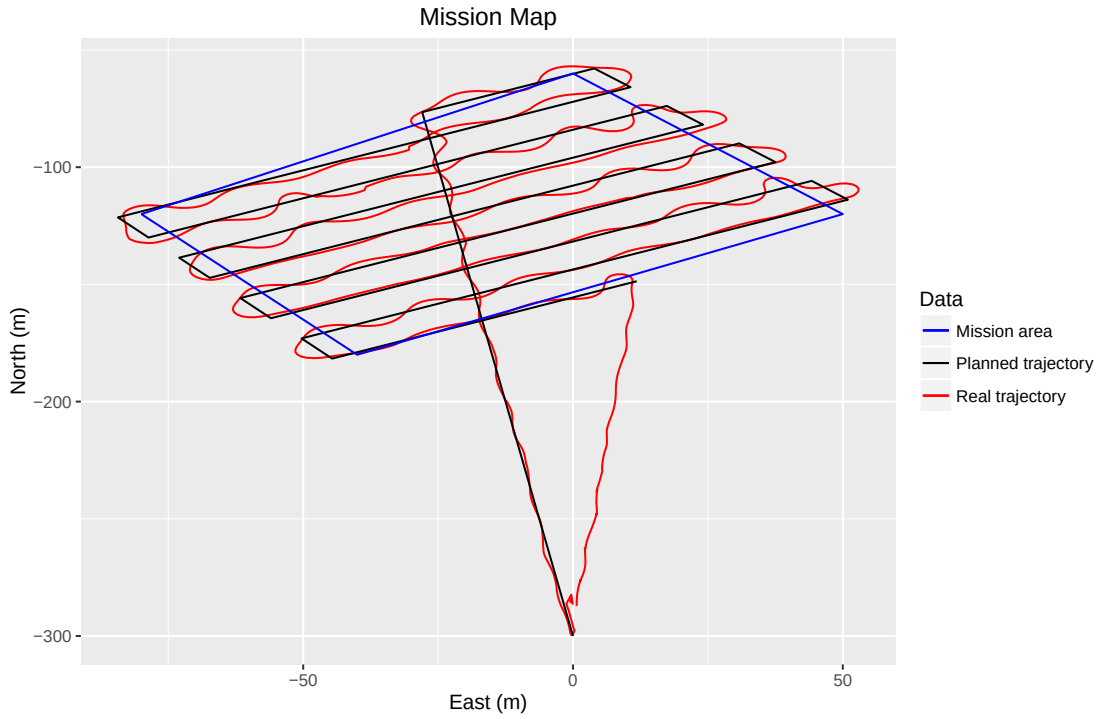


Figure 6.15: Plot of the trajectory generated by the offline planner and the real trajectory followed by the vehicle, for test case C.

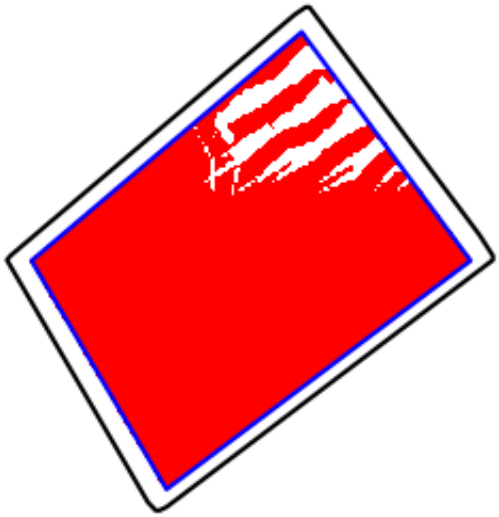
planned trajectory. Since there is no communication with the underwater vehicle, there was no other choice then to let it finish the mission. Afterwards, a problem with the compass was detected, as it was descalibrated. During the field trial, the only way to minimize the impact of the problem was to add an offset to the output and try to operate. The planned and real trajectories are represented in figure 6.18.

The replanner reported and average of 45% POD over the search area a map error of 10.1%. In reality, the seafloor is a bit deeper then expected.

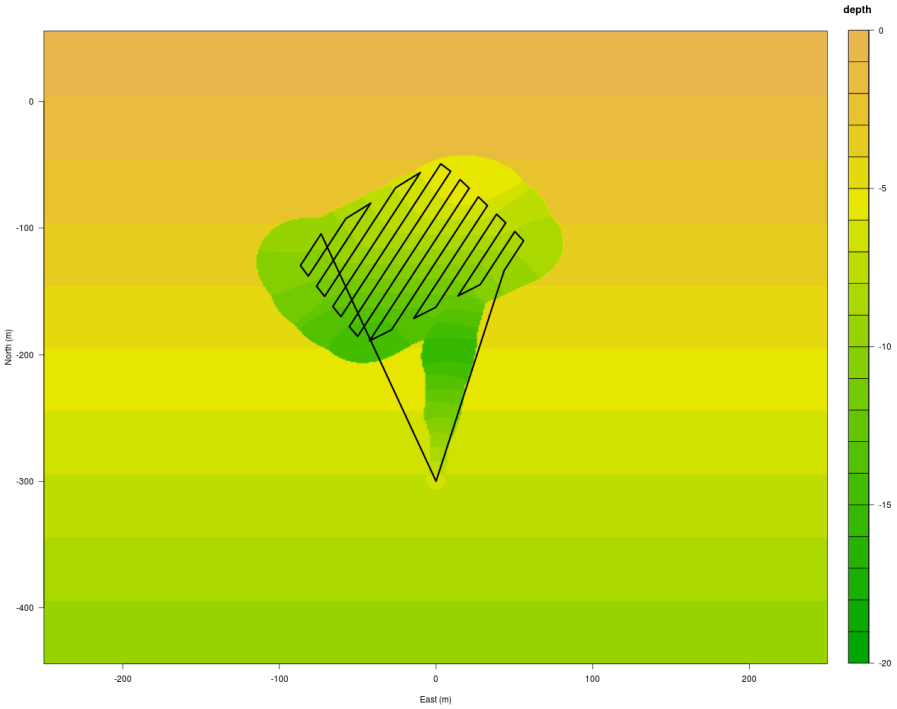
Although with a silhouette coefficient of 1.00, the identified cluster was well defined around the area not surveyed by the vehicle, as can be seen in figure 6.19b. The performance of the clustering algorithm is a bit questionable. When merging the clusters using single link similarity, clusters from both sides are joined together due to the small distance in the mid-bottom section. Such a situation never arose during simulation, where there was a hole in the middle of the operating area. In such a situation it seems advantageous to somehow detect the hole and don't allow the clusters from each side to merge.

The replanner required 36 seconds to replan the mission according to the previous performance and acquired data and produced a solution with 97% POD that required 22 minutes and 45 seconds to cover the area. The new trajectory can be seen in figure 6.19.

In any case, both planners behaved as seen during the simulations and were able to provide resource efficient solutions in every situation. In a real search operation though, where the sonar data is being actually processed during the mission, much of the imagery from the sonar might



(a) Identified clusters, showing area that was not covered with desired level of performance.



(b) Planned trajectory using online algorithm and updated map.

Figure 6.16: Output of the online planner for test case C.

be unusable if the vehicle follows a path with excessive curvature as was the case here. In our case, the excessive curvature was caused by the smoothing algorithm used for post-processing the trajectory since the raw navigation data is actually much more rectilinear. Anyway, this indicates other area for improvement. The trajectory evaluation algorithm should analyse the trajectory and exclude segments with too much curvature from the detection performance evaluation in order to

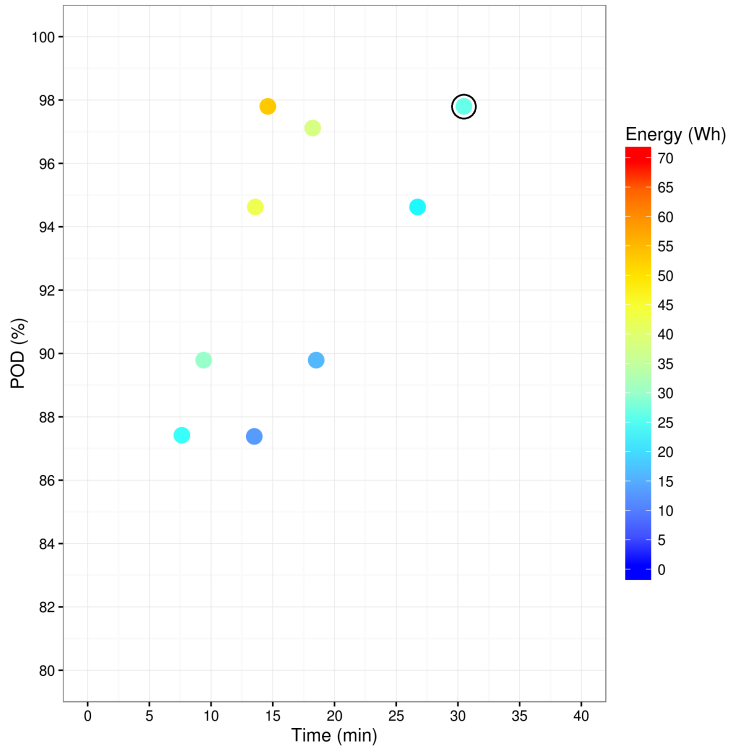


Figure 6.17: Pareto Front obtained by the offline planner considering the terrain from figure 6.9a. The chosen solution is is identified by the black circumference. This solution needed nearly double the amount of time when compared to the orange solution, but in return spent half of the energy. In fact, the only difference in terms of parameters was the velocity, with the chosen solution using a more conservative one.

not falsely skew the results.

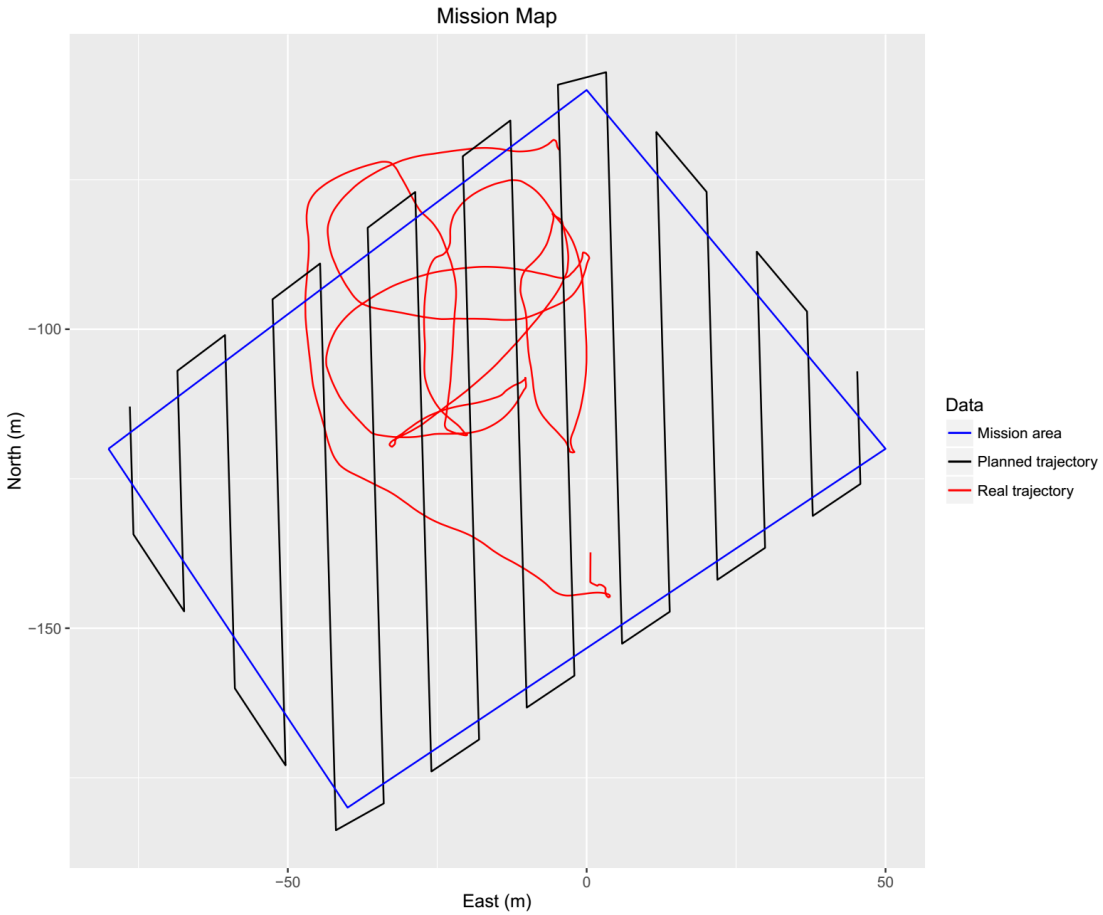
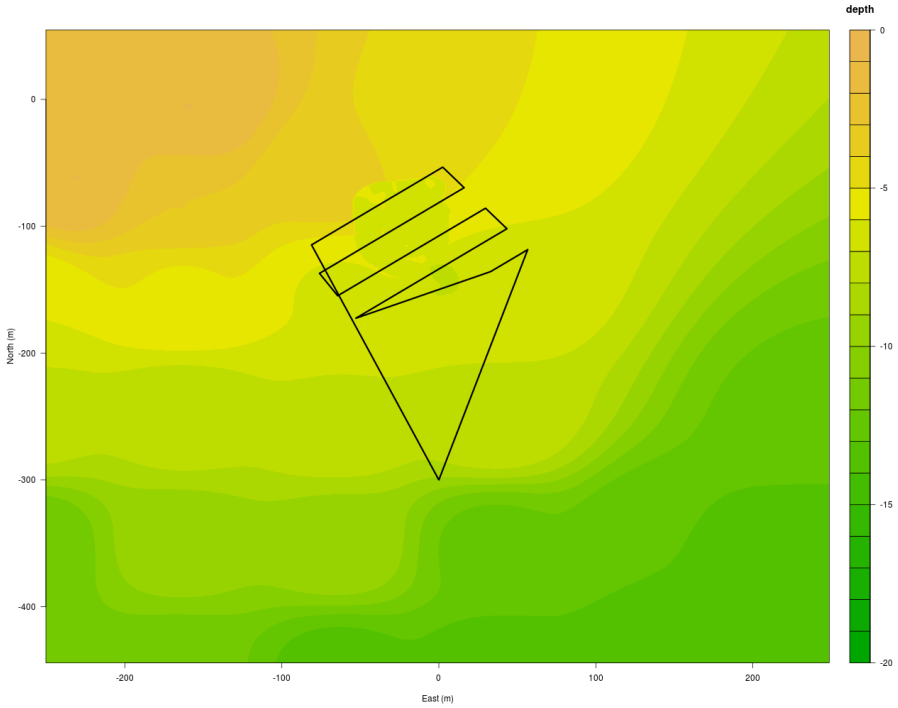


Figure 6.18: Plot of the trajectory generated by the offline planner and the real trajectory followed by the vehicle, for field trial.

(a) Identified clusters, showing area that was not covered with desired level of performance.



(b) Planned trajectory using online algorithm and updated map.

Figure 6.19: Output of the online planner for the field trial.

Chapter 7

Incorporation of uncertainty

7.1 Introduction

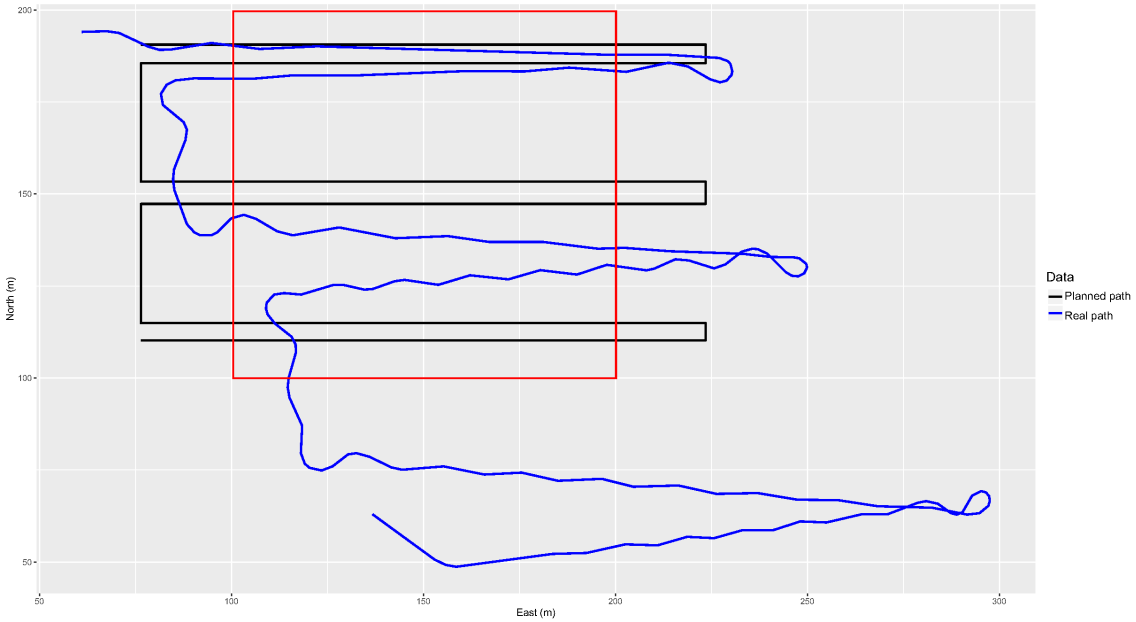
This chapter presents an algorithm[1] to improve coverage performance, accounting for navigation errors along mission execution and adapting the planned path in response. When covering an area using an AUV, it may not be possible to guarantee that all the area is covered since the path actually followed by the vehicle may deviate from the path planned by the coverage algorithm. The real path can deviate from the planned path due to:

- the existence of external disturbances such as sea currents and bad weather;
- the vehicle state estimate being uncertain and bounded or unbounded errors may exist;
- the performance of the motion controller not being good enough.

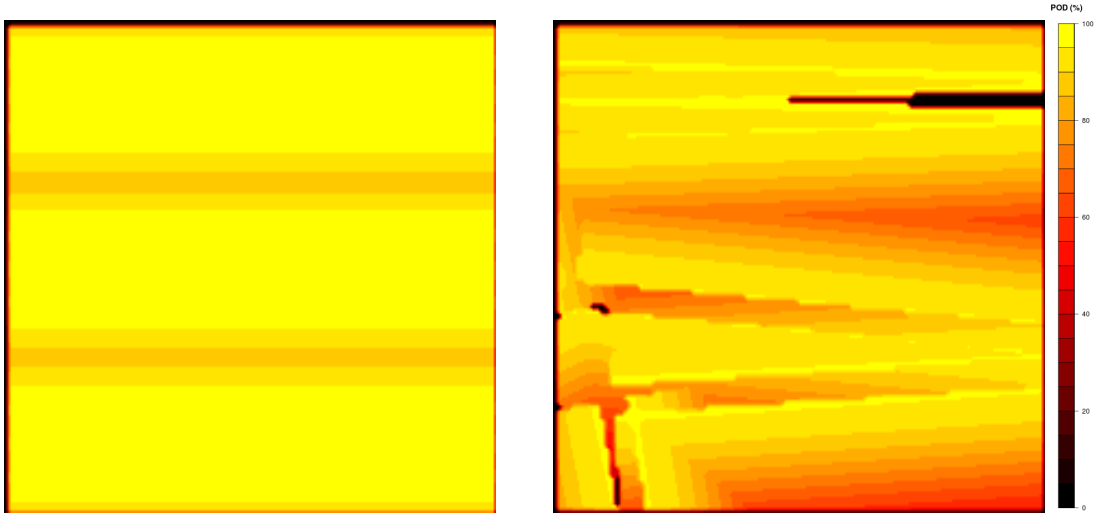
State estimation techniques commonly used in navigation systems, such as the Kalman filter, model sensor noise and produce estimates of the state prediction error. Hence, a methodology is required to incorporate this knowledge of uncertainty (which is already available) in the planning stage, compensating for the limitations of the navigation system. Ignoring the uncertainty that accompanies probabilistic state estimates during planning can lead to a costly replanning stage where there is a need to cover all of the operating area (in the worst case) due to the existence of small gaps between parallel tracks where coverage performance is below what is required. Potentially, it can even lead to planning or even mission failure if the vehicle is not able to georeference acquired data due to the limited capability of smoothing algorithms to reduce past uncertainty. For example, in cases where there is not complete knowledge about the environment, mapping algorithms may underperform when obtaining an inaccurate updated representation of the environment.

Consider figure 7.1a which displays the outcome of a simulation where the AUV was trying to cover an area with a sidescan sonar. The performance of a low cost inertial measurement unit (IMU) with a high drifting bias is being simulated, chosen specifically for demonstrating the problem under study. Figure 7.1b shows the estimated performance for the planned path.

[1]The work presented in this chapter was published in (Abreu et al., 2017).



(a) Plot of the planned path (in black, planning area in red) and the actual path followed by the vehicle (in blue).



(b) Expected detection performance.

(c) Real detection performance.

Figure 7.1: Effect of the navigation system's uncertainty in detection performance.

Figure 7.1c shows the real performance given the position of the vehicle during the simulated mission. When determining the proper MCM coverage plan, one is usually constrained by the time to complete the operation and the risk to which vehicles and personnel may be subjected to. Thus, the scenario described above represents an unacceptable risk to units involved in an MCM operation and clearly needs to be accounted for.

An algorithm is needed to improve coverage performance, accounting for navigation errors along mission execution and adapting the planned path in response. There are several measures of mission performance, such as coverage factor, POD or probability of success (POS). Each of these measures can be considered appropriate for the application. In particular, the POD measure of

performance is suggested (calculated using a lateral range curve as a function of range) together with the definition of a minimum level of performance at each point in the map. This would indirectly force complete coverage of the operating area while ensuring a lower bound on detection performance.

Aiming to bridge this gap, a CPP technique is presented which takes into account the robot's motion and sensing uncertainty and tries to minimize this uncertainty along the planned path. The objective is to plan paths, using a state prediction error model as input, to reduce as much uncertainty as possible and to minimize the extra path length (swath overlap) while satisfying mission feasibility constraints. In the end a modified lawn-mower search pattern is produced to accomplish these objectives. In particular, the decay in detection performance over the discretized workspace is assessed by projecting the expected variation in range through the coverage sensor lateral range curve. An iterative algorithm based on simulated annealing is proposed to optimize track distance. The optimization algorithm simulates the uncertainty throughout the path and varies track spacing in order to ensure that the detection performance is above a given threshold. Additionally, whenever the uncertainty after a parallel track exceeds a user-provided threshold, a directive for getting a navigation update is inserted, seeking to reduce uncertainty and to avoid creating a mission plan that would take too long to execute due to the constant (growing) swath overlap.

The proposed approach is similar to the one presented by Paull et al. (Paull et al., 2014) but our algorithm anticipates what will be the best moments for bringing the vehicle to surface to ensure a bounded position error. The algorithm also considers time and energy constraints that may influence the planned trajectory as path overlap is increased to account for uncertainty. Additionally, the assumption frequently seen in coverage applications where two observations of the same target are considered independent is challenged. Modern research involving the calculation of the cumulative detection performance in different case studies assume complete independence, but there are some problems with this approach (Carnes, 1993). By performing coverage on the same environment, with the same system and with increasing uncertainty in position it may be wise to consider that there is an indeterminate amount of correlation between observations. Therefore, an alternative approach will be presented and the real impact on detection performance estimation will be assessed.

7.2 Navigation system's uncertainty

Accurate navigation information is essential for achieving the expected mission performance, for safe vehicle operation and recovery. For the data acquired by an AUV to be of value, the location from which the data has been acquired must be accurately known. Current practice in AUV navigation is focused on dead-reckoning, INS, acoustic systems and terrain aided techniques. Next, a brief description of the errors involved in these systems is given.

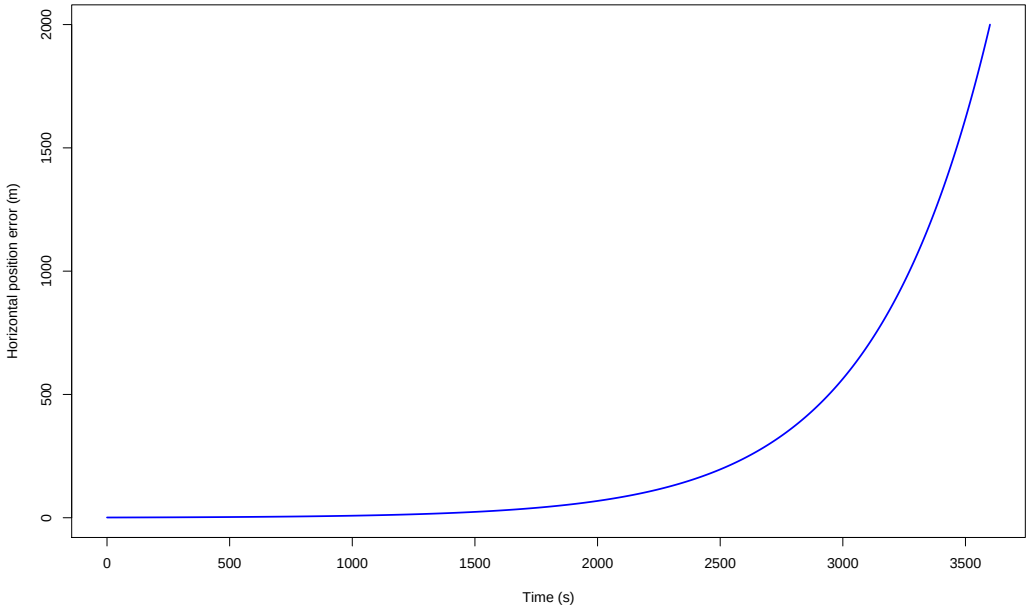


Figure 7.2: Horizontal plane error model for a modern strap-down INS.

7.2.1 Unbounded error

The problem with relying on inertial navigation is that the position error increases without bound as the distance travelled by the vehicle increases. The rate of increase will be dependent on currents, the vehicle speed, the quality of dead reckoning algorithm and other sensor data. GPS can provide an accurate position update provided the vehicle is able to return to surface periodically for a position fix. The maximum vehicle travel time between surfacings for a position update will be determined by the navigation system's accuracy, as described in figure 7.2. Systems integrating a poor quality IMU will face an unacceptably high frequency of surfacing. Surfacing always comes with a cost, especially on extra mission execution time. For deep water applications, the time and energy needed by an AUV for transiting to the surface can be very unfavourable. Also, vehicles operating close to important shipping routes are in considerable danger of collision with surface vessels if they need to frequently return to surface for position fixes. Position drift rates for high quality IMU units (unaided INS) are on the order of several kilometres per hour. Integrating a DVL sensor in the INS can greatly improve performance with drift rates ranging from 0.01% to 0.1% of the distance travelled depending on the format of paths.

7.2.2 Bounded error

Acoustic transponders can be used as beacons to guide the AUV without the need for resurfacing. Long baseline (LBL) navigation systems provide position accuracy of a few meters with a maximum range on the order of a few kilometres. Errors in acoustic systems come from several sources. The main sources of error are: errors in the assumed array geometry and errors in the

assumed sound speed profile. Positioning error comes from assuming incorrect positions of the array beacons. Reflection or multipath errors will result in erroneous time-of-flight values and thus wrong position fixes. Typically, LBL works well in deep water. When operating in shallower water, complex propagation effects become relevant and increase the frequency of bad position fixes. Even if the sound speed profile is known at the start of an AUV mission, the acoustic propagation environment can change during the mission (due to thermoclines, etc).

7.3 Methodology

Aiming to bridge this gap, a CPP technique is presented which takes into account the robot's motion and sensing uncertainty and tries to minimize this uncertainty along the planned path. The objective is to plan paths, using a state prediction error model as input, to reduce as much uncertainty as possible and to minimize the extra path length (swath overlap) while satisfying mission feasibility constraints. Here a modified lawn-mower search pattern is used to accomplish these objectives. In particular, the decay in detection performance over the discretized workspace is assessed by projecting the expected variation in range through the coverage sensor lateral range curve. An iterative algorithm based on simulated annealing is also proposed to optimize track distance. This is an optimization algorithm that involves the simulation of uncertainty throughout the path and the implementation of a feedback policy that ensures that the detection performance will above a certain threshold. This approach has the distinct advantage that it can be easily added to existent coverage path planning algorithms as an add-on, optimizing existent plans before creating the mission directives.

7.3.1 Uncertainty estimation

The uncertainty in the state estimate provided by the AUV's navigation system when performing a coverage mission may cause the vehicle to deviate from the planned trajectory. This means that when traversing straight parallel transects, the vehicle may actually be closer or further away from a given point located between two transects, as depicted in figure 7.3.

The component of uncertainty in position, perpendicular to the transects, can be interpreted as the uncertainty on effective distance to that fixed point on the workspace. Thus, range is a random variable R. This variation in range to a given cell will result in a variation in detection performance. This means that detection performance on each cell of the workspace will now be a random variable Y with a given probability distribution function (PDF) instead of a fixed quantity. It is assumed that the PDF of R, f_R, can be approximated by a Gaussian function:

$$f_R(r) = \frac{1}{\sqrt{2\pi\sigma_R}} e^{-\frac{(x-\mu_R)^2}{2\sigma_R^2}} \tag{7.1}$$

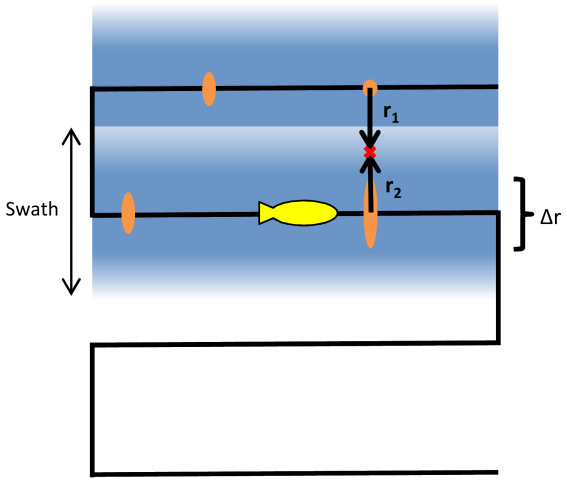


Figure 7.3: Example of a lawnmower trajectory followed by the AUV. As the vehicle travels along the path, the uncertainty of the navigation system's estimates (in orange) increases. The distance to a fixed point (in red) cannot be considered static, from a planning perspective, given the uncertainty in position.

Then the PDF of Y, f_Y, is obtained by applying the transformation technique for continuous random variables (Mood, 1950):

$$f_Y(y) = \left| \left(\frac{d}{dy} l^{-1}(y) \right) \right| f_R(l^{-1}(y)) \tag{7.2}$$

where $l(r)$, which is the LRC function used to characterize the detection system's performance, is a one-to-one function and $l^{-1}(y)$ is its inverse. For this study, the inverse cube model is considered an appropriate function to represent the LRC and apply this methodology:

$$l(r) = 1 - e^{-\frac{c}{r^2}} \tag{7.3}$$

where c is a constant that depends on the environment where the vehicle is operating and its velocity. By applying this principle, it is possible to derive the PDF and the CDF of Y, respectively:

$$f_Y(y) = \frac{\sqrt{2} \cdot 5^{\frac{3}{2}} \left| \frac{1}{\sqrt{\sigma_R}} \right| \left| \frac{1}{\sqrt{-\frac{1}{\ln(1-y)}}} \right| \mathrm{e}^{-\frac{\left(4 \cdot 5^{\frac{3}{2}} \sqrt{-\frac{1}{\ln(1-y)}} - \mu_R\right)^2}{2\sigma_R^2}}}{\sqrt{\pi} \ln^2(1-y) \, |y-1|} \tag{7.4}$$

$$F_Y(y) = \frac{\sigma_R \left| \frac{1}{\sqrt{\sigma_R}} \right| \mathrm{erf}\left(\frac{\sqrt{c}\sqrt{-\frac{1}{\ln(1-y)}} - \mu_R}{\sqrt{2}\sigma_R} \right) (y-1)}{2 \left| \frac{1}{\sqrt{-\frac{1}{\ln(1-y)}}} \right| \sqrt{-\frac{1}{\ln(1-y)}} \, |y-1|} \tag{7.5}$$

This was the process for estimating the detection performance distribution due to a single sonar measurement. As the vehicle may cover the same area twice with the sonar by traversing consecutive path transects, individual measurements need to be combined. For instance, in minehunting operations, where considerable risk is involved, the cost of overestimating the detection performance may be much higher than the cost of underestimating it. Hence, it should be beneficial to opt for a conservative estimate. Conservative estimates are obtained by assuming the existence of complete correlation, thus by choosing the higher of the individual estimates (*max* operator). When combining the observations, one can assume dependence or independence, being the latter frequently chosen as it simplifies the procedure of calculating a PDF. Both approaches will be compared by also estimating a PDF assuming dependence and assessing the impact on detection performance estimation.

The combined detection performance of two measurements Y_1 and Y_2 is given by:

$$Z = max(Y_1, Y_2) \tag{7.6}$$

For two independent random variables Y_1 and Y_2 , Z's CDF and PDF can be calculated with the following expression, respectively (Athanasios, 1965):

$$F_Z(z) = F_{Y_1}(z)F_{Y_2}(z) \tag{7.7}$$

$$f_Z(z) = f_{Y_1}(z)F_{Y_2}(z) + f_{Y_2}(z)F_{Y_1}(z) \tag{7.8}$$

In a real world situation, it is difficult to assess the degree of correlation between the two observations. If such correlation exists, the true distribution for the combined detection performance Z should be derived, if possible. However, due to the complexity of the mathematical functions, derivation of the equation is difficult. In such cases, Monte Carlo simulation is a suitable tool to estimate the real distribution function of Z. This said, the method was applied and a total of 100000 simulations were performed in order to obtain a precise estimate of F_Z. Figure 7.4 presents a visual comparison between both approaches. It is possible to see that the first approach is a little conservative until halfway through the mission, as expected. But surprisingly, from there onwards, the first approach became increasingly pessimistic as time, and localization uncertainty as a consequence, increased. Such a performance supports the argument that one cannot just assume independence between observations without assessing the true relationship between the variables.

In our case, if the mission planner considered independence blindingly, then it would bring the parallel tracks excessively together to compensate for the growing uncertainty and that would increase mission time and energy expenditure beyond what was really required. Unfortunately, given the difficulty in deriving the real F_Z and the time needed to perform the Monte Carlo simulation, there is no alternative to considering independence between the observations. To mitigate the effect on the calculation of the probabilistic measure of detection performance, an upper bound on time between surfacing manoeuvres is established to reduce localization uncertainty. In this ex-

periment, assuming a navigation error model as depicted in figure 7.2 and by analysing the maps on figure 7.4, a time interval of 575 seconds was chosen as the difference between the approaches was negligible up to this point.

7.3.2 Uncertainty reduction procedure

As a means to minimize the number of gaps in coverage resultant from uncertainty in the state's estimates, our local optimization algorithm will adjust the position of the parallel transects in order to guarantee that the probability of the detection performance Z being above a minimum value D_{min} is close to a predefined certainty level C:

Consider the following assumptions:

- operating area A;
- N tracks, parallel in projection to horizontal plane;
- each track i marks the end of a subarea a_i, so there area $N+1$ subareas;
- a variance threshold, V_{max}, that represents maximum variance allowed at any time during the mission;
- a maximum overlap parameter, O_{max}, that represents the maximum overlap allowed between two consecutive swaths.

The inputs for this procedure are:

- lawnmower search pattern;
- topography map;
- navigation system's error function;
- LRC;
- minimum detection performance D_{min} and required certainty level C;
- available time and energy for executing the mission;
- V_{max};
- O_{max}.

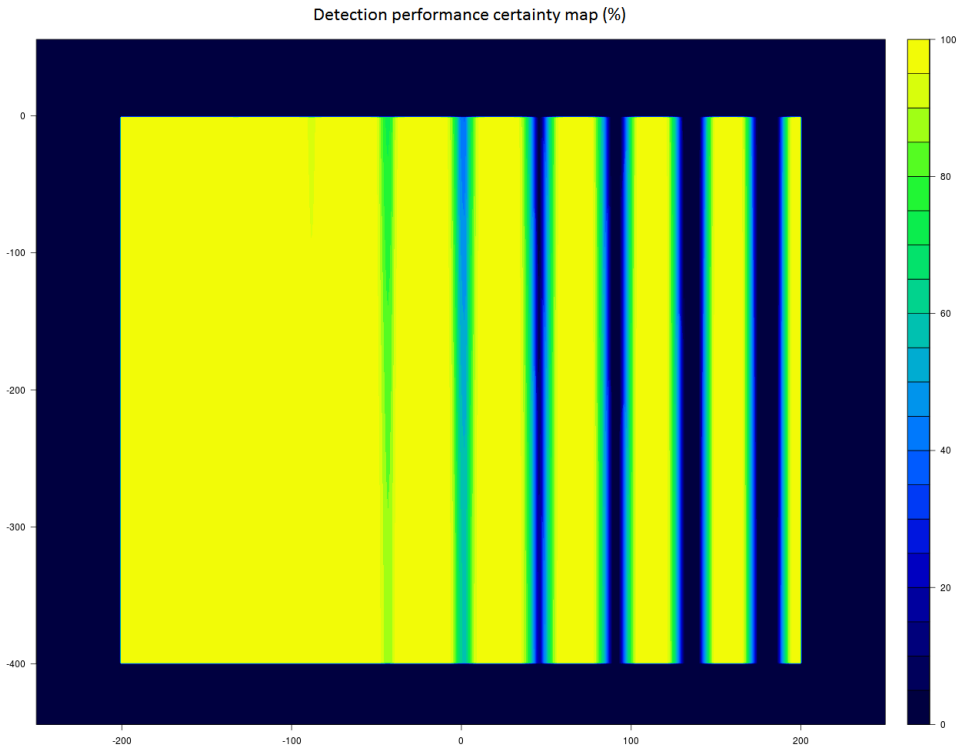
Then the outputs will be:

- optimized lawnmower search pattern;
- detection performance map;
- best time instants for surfacing for uncertainty reduction (if using INS).

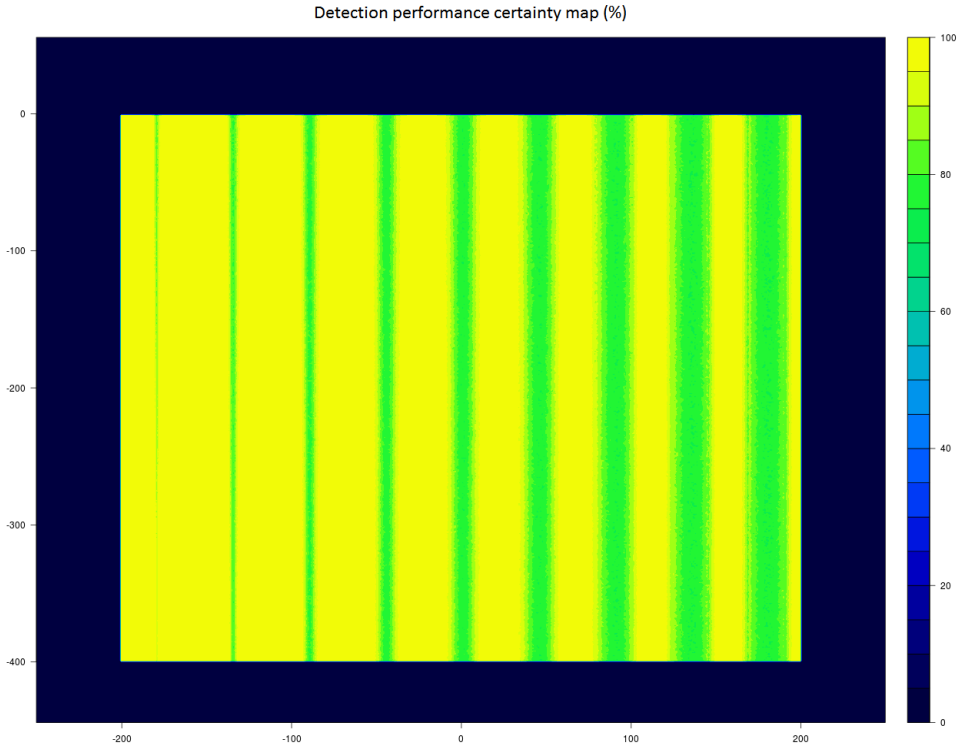
A high level description of the algorithm is presented in the flowchart depicted on figure 7.5. The first phase consists in analysing variance growth along the trajectory. Each time the variance surpasses V_{max} (which is the maximum squared deviation determined previously) a surfacing request is added to reduce uncertainty, preferably at the start of the track. Since variance is a function of time and the velocity of the vehicle is known, the distance between each of these points can be determined.

Next is the spacing optimization procedure. In order to increase the certainty that the detection performance requirements will be met, track spacing needs to be reduced. Here, the maximum overlap between two tracks needs to be considered. This was introduced to avoid creating paths that are too long. When energy and time constraints are broken after adding extra tracks, this parameter will be reduced in order to create smaller paths. Reducing overlap may also lead to the performance requirements not being met, in which case the solution which offers the best possible $P(Z \geq D_{min})$ throughout all operating area is returned. The algorithm for this procedure is an adaptation of the local optimization algorithm presented in section 5.4, where POD is replaced by $P(Z \geq D_{min})$ and the constraints introduced above are added.

Additionally, the location where each surfacing manoeuvre will take place can also be optimized. Since smoothing algorithms have limited performance and the error grows exponentially, it is beneficial to redistribute these locations in order to keep the error as low as possible at any time during the mission.



(a)



(b)

Figure 7.4: Detection performance certainty maps considering (a) independence and (b) dependence between observations.

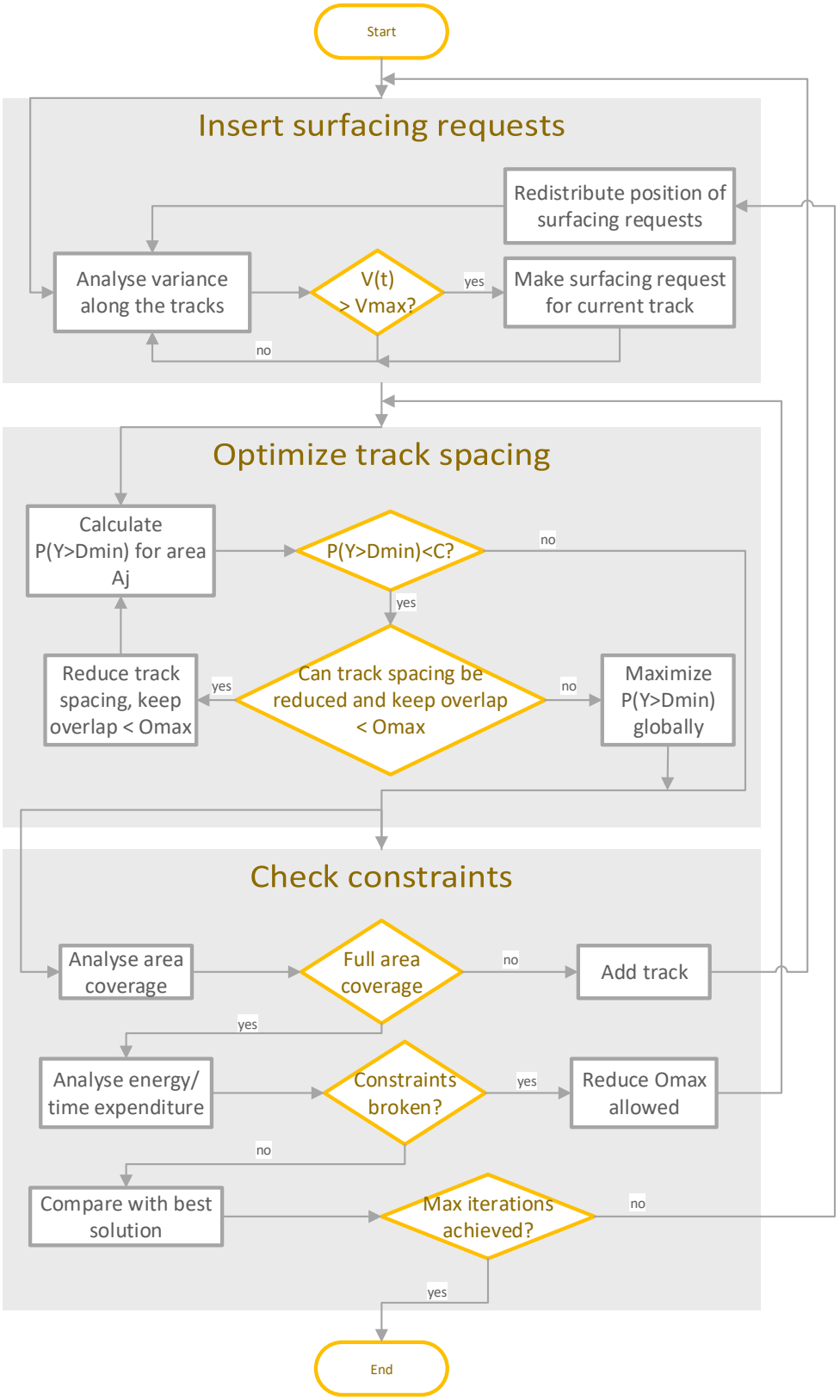


Figure 7.5: Flowchart for the uncertainty reduction algorithm.

Chapter 8

Conclusions and future work

8.1 Summary

This thesis deals with the design and development of a reliable and autonomous system necessary to plan and execute 3D surveys with different sensor payloads having search applications in mind. Different mission phases of an AUV operation were studied in detail and new techniques were proposed:

- Pre-mission (offline): advanced CPP algorithms take into account available elevation maps, the characteristics of the imaging sensors, the AUV's kinematic constraints and the uncertainty of the navigation system to generate survey trajectories;
- During-mission (online): replanning algorithms improve the offline planned paths given new information acquired during mission execution; this includes fusing multiple sensor data to create an updated map, identifying new survey areas and replanning the mission while accounting for the limited onboard processing power.

The offline approach is an multi-objective multi-stage approach combining a EA with SA for planning search operations in static 3D environment with known terrain. The algorithm maintains a diverse population of feasible solutions in order to explore the search space and uses SA to locally improve the best solutions found. The recombination and mutation operators used in our EA have been specifically designed concerning the simple individual representation that was considered to facilitate the evolutionary process and improve the quality of the found solutions. Our experiments showed that the proposed local optimization phase significantly helps to improve the performance of the algorithm, however at the cost of a higher computational time. The superiority of our informed algorithm, that used an ANN to guide the search, was demonstrated over an adaptation of SPEA 2 designed for our particular problem. The influence of several parameters was also assessed. First a good set of values for the parameters was found by executing a manual tuning procedure and then these were used as starting values for a proposed adaptive tuning procedure. It was found that the adaptive mutation magnitude contributes to a better performance. It was also exemplified what would be the typical output of the execution of our planning algorithm and

demonstrated the role that the decision maker may have to play when planning a search mission with an AUV. The proposed method for offline planning is compared to existent mission planners and the results demonstrate its superiority. These approaches use lawn-mower search patterns calculated before mission execution that can not be adapted while the vehicle is operating. When compared to the geometrical planner, it was also proved that, as complexity of the terrain increases, it delivers greater mission performance since it optimizes the trajectory to complex terrains.

The online mission replanning algorithm considers partially unknown terrains and allows the generation of new mission plans according to current mission performance. This approach reuses solutions from the evolutionary planner, considered as past experience, to speed up the replanning process. Several factors that influenced the choice of the best replanning strategy were studied and a global replanning algorithm that accounts for different scenarios was designed. The experiments showed that the online algorithm was very robust and able to successfully replan missions in varied scenarios, guaranteeing full area coverage while minimizing resource consumption. The major disadvantage of both offline and online algorithms is the amount of parameters that need to be configured. Although proper procedures were developed to test their impact in algorithm performance, one might still need to tweak them between operating scenarios. Processing time can also be seen as a disadvantage when compared to traditional approaches, but, as was shown, the increased performance in more complex scenarios justifies this approach. At the moment, the detection performance does not account for the shape of the trajectory. Detection performance, as calculated from the LRC, assumes that the searcher is navigating in a rectilinear path. One of the reasons for this is to maximize the quality of the sonar imagery (less distortion). Therefore, a good feature to develop in the future is analysing the curvature of the path and discard it (from the detection performance evaluation point of view) if if goes beyond a threshold.

Additionally, a CPP technique for search operations was described which takes into account the vehicle's position and detection performance uncertainties during mission execution and tries to minimize this uncertainty along the planned path. Such a technique is required in order to incorporate this knowledge of uncertainty in the offline planning stage, compensating for the limitations of the navigation system. It can be interpreted as an add-on to the offline algorithm. Accounting for uncertainty during planning reduces the need for a costly replanning stage, where full area coverage may be required (in the worst case) due to the existence of small gaps in coverage between the parallel tracks, and increases the quality of the acquired data. Results showed that the method was successful in minimizing uncertainty and selecting the best positions for performing the surfacing manoeuvres.

8.2 Future work

After working with robots for the last few years, one of the main conclusions we could make is that every subsystem is equally essential to the successful execution of a mission. There is no point in having the perfect mission planning system if the vehicle can't follow the planned trajectory with minimal deviation or if the detection system can't detect the objects seen by the sensors.

Therefore, we will prioritize improvements in the navigation system and in sonar data processing algorithms.

As for navigation system, we will focus on improving the INS algorithm for handling noise and bias and improve overall performance. We will also consider integrating other techniques such as Long baseline (LBL) navigation and simultaneous localization and mapping (SLAM). Recent advances in SLAM have greatly improved mobile robot localization, using statistical techniques and features of the terrain to correct the robot's pose estimation.

As for sonar data processing, one very important addition to the sonar simulation module is the ability to generate, in real-time, realistic sonar echo data. As explained earlier, the propagation of an acoustic beam is determined by a ray theory algorithm. A beam of rays is emitted and as each ray returns, the intensity of the reflection is added to a pixel. This value depends on the time elapsed since the acoustic pulse was transmitted. The contribution of each point of the terrain model to the estimated intensity at the transducer is determined by applying a seafloor scattering model. This synthetic images will allow object detection algorithms to assess, adapt and improve its performance in any environment since they use template matching algorithms for detecting the objects that are being searched. If appropriate assumptions are made, it would also be interesting to develop algorithms to convert the resulting backscatter images into an elevation map of the seafloor and applied this to real sonar data acquired during the mission.

Finally, other important extension is to develop a multi-vehicle version of the mission planning algorithm. For a small environment, it is more appropriate to use a single vehicle. For a large environment where multiple operating areas may be considered a multi-vehicle team could significantly shorten the time required to finish the operation. The problem here will be how to coordinate the team to perform the search mission. One hypothesis is to assume a distributed architecture, where the vehicles do not explicitly work together but cooperate by interacting separately with the environment. For example, each vehicle could be assigned to a given independent operating area. Since the effort of the team is not coordinated, the mission planning process is very inefficient since vehicles will have to wait for each other to complete the mission. Other hypothesis is to assume a centralized architecture, where a single vehicle or a basestation computer acts as the leader of the team. The vehicles upload acquired data and their current status to the leader as the mission progresses. The leader is then responsible for processing all the information and replanning the mission, coordinating the actions of all vehicles. This is the preferred approach as the leader might possess higher processing power capabilities needed for maintaining a centralized updated image of the DEM and for processing all acquired information.

8.3 Publications

Abreu, N. and A. Matos. Minehunting mission planning for autonomous underwater systems using evolutionary algorithms. *Unmanned Systems*, 2(04):323–349, 2014a.

Abreu, N. and A. Matos. Using evolutionary algorithms to plan automatic minehunting operations. In *Informatics in Control, Automation and Robotics (ICINCO), 2014 11th International Conference on*, volume 1, pages 228–235. IEEE, September 2014b.

Abreu, N., A. Matos, and N. Cruz. Accounting for uncertainty in search operations using AUVs. In *Underwater Technology (UT), 2017 IEEE*. IEEE, February 2017.

Abreu, N. and A. Matos. Case-based replanning of search missions using AUVs. In *Proc. IEEE OCEANS 2017 Aberdeen*. IEEE, June 2017.

8.4 Projects

This work was partially performed within the project "Increased autonomy for AUVs (mission planning and obstacle avoidance) - Mission Planning" managed by the European Defence Agency (EDA) under its Unmanned Maritime Systems (UMS) programme.

Bibliography

Aamodt, A. and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, 7(1):39–59, 1994.

Abreu, N. and A. Matos. Minehunting mission planning for autonomous underwater systems using evolutionary algorithms. *Unmanned Systems*, 2(04):323–349, 2014a.

Abreu, N. and A. Matos. Using evolutionary algorithms to plan automatic minehunting operations. In *Informatics in Control, Automation and Robotics (ICINCO), 2014 11th International Conference on*, volume 1, pages 228–235. IEEE, September 2014b.

Abreu, N. and A. Matos. Case-based replanning of search missions using AUVs. In *Proc. IEEE OCEANS 2017 Aberdeen*. IEEE, June 2017.

Abreu, N., A. Matos, and N. Cruz. Accounting for uncertainty in search operations using AUVs. In *Underwater Technology (UT), 2017 IEEE*. IEEE, February 2017.

Acar, E. and H. Choset. Sensor-based coverage of unknown environments: Incremental construction of morse decompositions. *International Journal of Robotics Research*, 21:345–366, 2002a.

Acar, E. U., H. Choset, A. A. Rizzi, P. N. Atkar, and D. Hull. Morse decompositions for coverage tasks. *International Journal of Robotics Research*, 21:331–344, 2002.

Acar, E. U. and H. Choset. Exploiting critical points to reduce positioning error for sensor-based navigation. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 4, pages 3831–3837. IEEE, 2002b.

Acar, E. U., H. Choset, Y. Zhang, and M. Schervish. Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods. *The International journal of robotics research*, 22(7-8):441–466, 2003.

Acar, E. U., H. Choset, and J. Y. Lee. Sensor-based coverage with extended range detectors. *Robotics, IEEE Transactions on*, 22(1):189–198, 2006.

Athanasios, P. *Probability, random variables, and stochastic processes*, page 141. McGraw-Hill, 1965.

B. Ferreira, A. M., M. Pinto and N. Cruz. Modeling and motion analysis of the mares autonomous underwater vehicle. In *MTS-IEEE Conference Oceans'09*, 2009.

Back, T., D. B. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation*. IOP Publishing Ltd., Bristol, UK, UK, 1st edition, 1997. ISBN 0750303921.

Bakker, B., M. Steingrover, R. Schouten, E. Nijhuis, and L. Kester. Cooperative multi-agent reinforcement learning of traffic lights. *ACM Transactions on Multimedia Computing, Communications, and Applications - TOMCCAP*, 01 2005.

Barrientos, A., J. Colorado, J. d. Cerro, A. Martinez, C. Rossi, D. Sanz, and J. Valente. Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots. *Journal of Field Robotics*, 28(5):667–689, 2011.

Barshan, B. and H. F. Durrant-Whyte. Inertial navigation systems for mobile robots. *Robotics and Automation, IEEE Transactions on*, 11(3):328–342, 1995.

Baylog, J. and T. Wettergren. Online determination of the potential benefit of path adaptation in undersea search. *IEEE J. Oceanic Eng.*, 39(1):165–178, Jan 2014.

Bays, M. J. *Stochastic Motion Planning for Applications in Subsea Survey and Area Protection*. PhD thesis, Virginia Polytechnic Institute and State University, 2012.

Beasley, J. E. and P. C. Chu. A genetic algorithm for the set covering problem. *European Journal of Operational Research*, 94(2):392–404, 1996.

Beom, H. and H. Cho. A sensor-based obstacle avoidance controller for a mobile robot using fuzzy logic and neural network. In *Intelligent Robots and Systems, 1992., Proceedings of the 1992 lEEE/RSJ International Conference on*, volume 2, pages 1470–1475. IEEE, 1992.

Box, G. E. Evolutionary operation: A method for increasing industrial productivity. *Applied statistics*, pages 81–101, 1957.

Branting, L. K. and D. W. Aha. Stratified case-based reasoning: Reusing hierarchical problem solving episodes. In *IJCAI*, pages 384–390. Citeseer, 1995.

Bremermann, H. J. Optimization through evolution and recombination. *Self-organizing systems*, 93:106, 1962.

Brooks, R. A. and T. Lozano-Perez. A subdivision algorithm in configuration space for findpath with rotation. *IEEE Transactions on Systems, Man, and Cybernetics*, (2):224–233, 1985.

Butler, Z. J., A. A. Rizzi, and R. L. Hollis. Contact sensor-based coverage of rectilinear environments. In *IEEE Int Intelligent Control/Intelligent Systems and Semiotics Symposium*, pages 266–271, 1999.

Canny, J. F. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, USA, 1988.

Carnes, J. Effective area searching for ground team members: Pod, critical spacing, and smart search techniques. Technical report, California OES and MRA report, 1993.

Choset, H. Coverage of known spaces: the boustrophedon cellular decomposition. *Autonomous Robots*, 9:247–253, 2000.

Choset, H. Coverage for robotics - a survey of recent results. In *Ann. Math. Artif. Intell.* SCITEPRESS, 2001.

Coit, D. W. and A. E. Smith. Penalty guided genetic search for reliability design optimization. *Computers and Industrial Engineering*, 30:895–904, 1996.

Connell, J. H. and S. Mahadevan, editors. *Robot Learning*. Kluwer Academic Publishers, Norwell, MA, USA, 1993. ISBN 0792393651.

Craik, F. I. and R. S. Lockhart. Levels of processing: A framework for memory research. *Journal of verbal learning and verbal behavior*, 11(6):671–684, 1972.

Cruz, N. A., A. C. Matos, R. M. Almeida, B. M. Ferreira, and N. Abreu. Trimares-a hybrid auv/rov for dam inspection. In *OCEANS'11 MTS/IEEE KONA*, pages 1–7. IEEE, 2011.

Darwin, C. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. John Murray, London, UK, 1859.

Das, J., K. Rajan, S. Frolov, F. Pyy, J. Ryany, D. Caron, and G. Sukhatme. Towards marine bloom trajectory prediction for auv mission planning. In *Int. Conf. Robot. Autom. (ICRA) 2010*, pages 4784–4790. IEEE, May 2010.

Deb, K. *Multi-objective optimization using evolutionary algorithms*. Wiley, UK, 2001.

Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE T. Evolut. Comput.*, 6(2):182–197, 2002.

Dempsey, P., S. Vaidya, and G. Cheng. The art of war: Innate and adaptive immune responses. *Cellular and Molecular Life Sciences CMLS*, 60(12):2604–2621, 2003.

Di Franco, C. and G. Buttazzo. Energy-aware coverage path planning of uavs. In *Autonomous Robot Systems and Competitions (ICARSC), 2015 IEEE International Conference on*, pages 111–117. IEEE, 2015.

Dogru, S. and L. Marques. Towards fully autonomous energy efficient coverage path planning for autonomous mobile robots on 3d terrain. In *Mobile Robots (ECMR), 2015 European Conference on*, pages 1–6. IEEE, 2015.

Douglas, D. H. and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

Duda, R. O., P. E. Hart, and D. G. Stork. *Pattern classification*. John Wiley & Sons, 1973.

Durrant-Whyte, H. Introduction to estimation and the kalman filter. *Australian Centre for Field Robotics*, 28(3):65–94, 2001.

Elfes, A. Occupancy grids: A stochastic spatial representation for active robot perception. In *Proceedings of the Sixth Conference on Uncertainty in AI*, volume 2929, 1990.

Fang, C. and S. Anstee. Coverage path planning for harbour seabed surveys using an autonomous underwater vehicle. In *Proc. IEEE OCEANS 2010 Sydney*, pages 1–8. IEEE, May 2010.

Fernández, J. J., M. Prats, P. J. Sanz, J. C. García, R. Marin, M. Robinson, D. Ribas, and P. Ridao. Grasping for the seabed: Developing a new underwater robot arm for shallow-water intervention. *IEEE Robotics & Automation Magazine*, 20(4):121–130, 2013.

Fournier, A., D. Fussell, and L. Carpenter. Computer rendering of stochastic models. *Communications of the ACM*, 25(6):371–384, 1982.

Friedberg, R., B. Dunham, and J. North. A learning machine: Part ii. *IBM Journal of Research and Development*, 3(7):282–287, 1959.

Friedberg, R. M. A learning machine: Part i. *IBM Journal of Research and Development*, 2(1): 2–13, 1958.

Fujimura, K. Path planning with multiple objectives. *Robot Autom. Mag.*, 3(1):33–38, Mar 1996.

Galceran, E. and M. Carreras. Efficient seabed coverage path planning for asvs and auvs. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 88–93. IEEE, 2012.

Galceran, E., S. Nagappa, M. Carreras, P. Ridao, and A. Palomer. Uncertainty-driven survey path planning for bathymetric mapping. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 6006–6012. IEEE, 2013.

Gelb, A. *Applied optimal estimation*. MIT press, 1974.

Geng, L., Y. Zhang, P. Wang, J. J. Wang, J. Y. Fuh, and S. Teo. Uav surveillance mission planning with gimbaled sensors. In *Control & Automation (ICCA), 11th IEEE International Conference on*, pages 320–325. IEEE, 2014.

Glasius, R., A. Komoda, and S. C. Gielen. Neural network dynamics for path planning and obstacle avoidance. *Neural Networks*, 8(1):125–133, 1995.

Glickman, M., J. Balthrop, and S. Forrest. A machine learning evaluation of an artificial immune system. *Evolutionary Computation*, 13(2):179–212, 2005.

Goel, T. S. and N. Stander. A study of convergence characteristics of multi-objective evolutionary algorithms. In *Proceedings of the 13th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2010.

Grasmueck, M., G. P. Eberli, D. A. Viggiano, T. Correa, G. Rathwell, and J. Luo. Autonomous underwater vehicle (auv) mapping reveals coral mound distribution, morphology, and oceanography in deep water of the straits of florida. *Geophysical Research Letters*, 33(23), 2006.

Haigh, K. Z. and J. R. Shewchuk. Geometric similarity metrics for case-based reasoning. In *Proc. of AAAI-94, Workshop on Case-Based Reasoning*, pages 182–187. Seattle, WA, 1994.

Hameed, I. Intelligent coverage path planning for agricultural robots and autonomous machines on three-dimensional terrain. *Journal of Intelligent & Robotic Systems*, 74(3-4):965–983, 2014.

Hebb, D. The organization of behavior, 1949.

Hert, S., S. Tiwari, and V. Lumelsky. A terrain-covering algorithm for an auv. In *Underwater Robots*, pages 17–45. Springer, 1996.

Holland, J. H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* Oxford, England: U Michigan Press, 1975.

Hollinger, G. A., U. Mitra, and G. S. Sukhatme. Active and adaptive dive planning for dense bathymetric mapping. In *Experimental Robotics*, pages 803–817. Springer, 2013.

Houts, S. E., S. M. Rock, and R. McEwen. Aggressive terrain following for motion-constrained auvs. In *Autonomous Underwater Vehicles (AUV), 2012 IEEE/OES*, pages 1–7. IEEE, 2012.

Huang, G. Learning capability and storage capacity of two-hidden-layer feedforward networks. *IEEE T. Neural Networ.*, 14(2):274–282, Mar 2003.

Itti, L., C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 20(11):1254–1259, 1998.

Jimenez, P. A., B. Shirinzadeh, A. Nicholson, and G. Alici. Optimal area covering using genetic algorithms. In *Advanced intelligent mechatronics, 2007 IEEE/ASME international conference on*, pages 1–5. IEEE, 2007.

Jin, J. and L. Tang. Coverage path planning on three-dimensional terrain for arable farming. *Journal of Field Robotics*, 28(3):424–440, 2011.

Johnson, D. S. and L. A. McGeoch. The traveling salesman problem: A case study in local optimization. *Local search in combinatorial optimization*, 1:215–310, 1997.

Johnson-Roberson, M., O. Pizarro, S. B. Williams, and I. Mahon. Generation and visualization of large-scale three-dimensional reconstructions from underwater robotic surveys. *Journal of Field Robotics*, 27(1):21–51, 2010.

Kavraki, L. *Probabilistic Roadmaps for Path Planning in High-dimensional Configuration Spaces*. Number no. 1519 in Computer Science Department: Report STAN-CS. Department of Computer Science, Stanford University, 1994.

Kermorgant, H. Architecture of auv systems for harbour protection and mine countermeasure. In *Europe Oceans 2005*, volume 2, pages 1400–1405. IEEE, 2005.

Kira, Z. and R. Arkin. Forgetting bad behavior: Memory management for case-based navigation. In *RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2004.

Kohl, N. and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 3, pages 2619–2624. IEEE, 2004.

Koopman, B. *Search and Screening: General Principles with Historical Applications*. The Military Operations Research Society, Virginia, USA, 1946.

Kruusmaa, M. Global level path planning for mobile robots in dynamic environments. *Journal of Intelligent and Robotic Systems*, 38(1):55–83, 2003.

Kuhlman, M. J., P. Svec, K. N. Kaipa, D. Sofge, and S. K. Gupta. Physics-aware informative coverage planning for autonomous vehicles. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 4741–4746. IEEE, 2014.

Latombe, J.-C. *Robot Motion Planning*. The Springer International Series in Engineering and Computer Science. Springer, 1990.

Lavalle, S. M. Rapidly-exploring random trees: A new tool for path planning. Technical report, Computer Science Dept., Iowa State University, 1998.

LaValle, S. M. *Planning algorithms*. Cambridge university press, 2006.

Leake, D. B. and D. C. Wilson. When experience is wrong: Examining cbr for changing tasks and environments. In *Case-Based Reasoning Research and Development*, pages 218–232. Springer, 1999.

Lee, T.-K., S.-H. Baek, Y.-H. Choi, and S.-Y. Oh. Smooth coverage path planning and control of mobile robots based on high-resolution grid map representation. *Robotics and Autonomous Systems*, 59(10):801–812, 2011.

Lee, T.-S. and B. H. Lee. A new hybrid terrain coverage method for underwater robotic exploration. *Journal of Marine Science and Technology*, 19(1):75–89, 2014.

Lee, T.-S., J.-S. Choi, J.-H. Lee, and B.-H. Lee. 3-d terrain covering and map building algorithm for an auv. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 4420–4425. IEEE, 2009.

Lee, W.-C. and J. Krumm. Trajectory preprocessing. In *Computing with spatial trajectories*, pages 3–33. Springer, 2011.

Likhachev, M., M. Kaess, and R. C. Arkin. Learning behavioral parameterization using spatio-temporal case-based reasoning. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1282–1289. IEEE, 2002.

Lumelsky, V. J. and T. Skewis. Incorporating range sensing in the robot navigation function. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(5):1058–1069, 1990.

Lumelsky, V. J. and A. A. Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2(1-4):403–430, 1987.

Mannadiar, R. and I. Rekleitis. Optimal coverage of a known arbitrary environment. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 5525–5530. IEEE, 2010.

Martin, O., S. W. Otto, and E. W. Felten. *Large-step Markov chains for the traveling salesman problem*. Oregon Graduate Institute of Science and Technology, Department of Computer Science and Engineering, 1991.

Martin, O., S. W. Otto, and E. W. Felten. Large-step markov chains for the tsp incorporating local search heuristics. *Oper. Res. Lett.*, 11(4):219–224, 1992.

Maza, I. and A. Ollero. Multiple uav cooperative searching operation using polygon area decomposition and efficient coverage algorithms. In *Distributed Autonomous Robotic Systems 6*, pages 221–230. Springer, 2007.

McCulloch, W. S. and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):18–28, 1943.

Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21(6):1087–1092, 1953.

Miller, G. S. The definition and rendering of terrain maps. In *ACM SIGGRAPH Computer Graphics*, volume 20, pages 39–48. ACM, 1986.

Miller, I. and M. Campbell. A mixture-model based algorithm for real-time terrain estimation. In *The 2005 DARPA Grand Challenge*, pages 407–436. Springer, 2007.

Miller, W. T., P. J. Werbos, and R. S. Sutton. *Neural networks for control*. MIT press, 1990.

Mood, A. M. *Introduction to the Theory of Statistics.*, page 200. McGraw-hill, 1950.

Moore, A. W. *Efficient memory-based learning for robot control.* PhD thesis, University of Cambridge, 1990.

Morin, M., I. Abi-Zeid, Y. Petillot, and C.-G. Quimper. A hybrid algorithm for coverage path planning with imperfect sensors. In *IEEE Int. Conf. Int. Robot (IROS) 2013*, pages 5988–5993, Nov 2013.

Nash, L., G. Hover, and R. Burns. Additional analyses of probability of detection (pod) in search and rescue (sar) project data. Technical report, DTIC Document, 1982.

Noice, H. and T. Noice. What studies of actors and acting can tell us about memory and cognitive functioning. *Current Directions in Psychological Science*, 15(1):14–18, 2006.

Oksanen, T. and A. Visala. Coverage path planning algorithms for agricultural field machines. *Journal of Field Robotics*, 26(8):651–668, 2009.

Pagac, D., E. M. Nebot, and H. Durrant-Whyte. An evidential approach to map-building for autonomous vehicles. *IEEE Transactions on Robotics and Automation*, 14(4):623–629, 1998.

Paull, L., S. Saeedi, M. Seto, and H. Li. Sensor-driven online coverage planning for autonomous underwater vehicles. *Mechatronics, IEEE/ASME Transactions on*, 18(6):1827–1838, 2013.

Paull, L., M. Seto, and H. Li. Area coverage planning that accounts for pose uncertainty with an auv seabed surveying application. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 6592–6599. IEEE, 2014.

Peters, J., K. Mülling, J. Kober, D. Nguyen-Tuong, and O. Krömer. Towards motor skill learning for robotics. In *Robotics Research*, pages 469–482. Springer, 2011.

Petillot, Y., S. Reed, and J. Bell. Real time auv pipeline detection and tracking using side scan sonar and multi-beam echo-sounder. In *OCEANS'02 MTS/IEEE*, volume 1, pages 217–222. IEEE, 2002.

Pinto, D. and B. Barán. Solving multiobjective multicast routing problem with a new ant colony optimisation approach. In *In: Proceedings of the 3rd international IFIP/ACM Latin American conference on Networking*, pages 11–19, 2005.

Pomerleau, D. A. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991.

Prior, M. Simple sonar modelling for diver detection. Technical report, NATO Undersea Research Centre, 2006.

Rauch, H. E., C. Striebel, and F. Tung. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8):1445–1450, 1965.

Rechenberg, I. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Problemata, 15. Frommann-Holzboog, 1973.

Riedmiller, M., A. Moore, and J. Schneider. Reinforcement learning for cooperating and communicating reactive agents in electrical power grids. In *Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, pages 137–149. Springer, 2000.

Rollins, C. A cumulative detection probability method. In *28th Navy Symp. on Underwater Acoustics*, pages 121–133, 1970.

Roman, C. and H. Singh. A self-consistent bathymetric mapping algorithm. *Journal of Field Robotics*, 24(1-2):23–50, 2007.

Russell, S. and P. Norvig. Artificial intelligence: A modern approach. *Artificial Intelligence*, 2003.

Sadrpour, A., J. Jin, and A. G. Ulsoy. Mission energy prediction for unmanned ground vehicles. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2229–2234, May 2012.

Schaal, S., C. G. Atkeson, and S. Vijayakumar. Scalable techniques from nonparametric statistics for real time robot learning. *Applied Intelligence*, 17(1):49–60, 2002.

Schwefel, H. *Evolutionsstrategie und numerische Optimierung*. Technische Universität Berlin, 1975.

Shepard, D. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM National Conference*, ACM '68, pages 517–524, New York, NY, USA, 1968. ACM.

Shnaps, I. and E. Rimon. Online coverage of planar environments by a battery powered autonomous mobile robot. *IEEE Transactions on Automation Science and Engineering*, 13(2): 425–436, April 2016.

Smart, W. D. and L. P. Kaelbling. Effective reinforcement learning for mobile robots. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 4, pages 3404–3410. IEEE, 2002.

Smyth, B. and M. Keane. Remembering to forget: a competence preserving deletion policy for case-based reasoning system. In *Proc. 14th Int. Joint Conf. Artificial Intelligence*, pages 377–382. Morgan-Kaufmann, 1995.

Soltero, D. E., M. Schwager, and D. Rus. Generating informative paths for persistent sensing in unknown environments. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2172–2179. IEEE, 2012.

Stack, J. R. and C. M. Smith. Combining random and data-driven coverage planning for underwater mine detection. In *OCEANS 2003. Proceedings*, volume 5, pages 2463–2468. IEEE, 2003.

Stephan, V., K. Debes, H.-M. Gross, F. Wintrich, and H. Wintrich. A reinforcement learning based neural multiagent system for control of a combustion process. In *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, volume 6, pages 217–222. IEEE, 2000.

Stützle, T. G. *Local search algorithms for combinatorial problems: analysis, improvements, and new applications*, volume 220. Infix Sankt Augustin, Germany, 1999.

Sutton, R. S. and A. G. Barto. *Introduction to reinforcement learning*, volume 135. MIT Press Cambridge, 1998.

Thomaz, A. L. and C. Breazeal. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence*, 172(6):716–737, 2008.

Torres, M., D. A. Pelta, J. L. Verdegay, and J. C. Torres. Coverage path planning with unmanned aerial vehicles for 3d terrain reconstruction. *Expert Systems with Applications*, 55:441–451, 2016.

Vasudevan, C. and K. Ganesan. Case-based path planning for autonomous underwater vehicles. In *Intelligent Control, 1994., Proceedings of the 1994 IEEE International Symposium on*, pages 160–165. IEEE, 1994.

Welch, G. and G. Bishop. An introduction to the kalman filter. *Proceedings of the Siggraph Course, Los Angeles*, 2001.

Wiering, M. et al. Multi-agent reinforcement learning for traffic light control. In *ICML*, pages 1151–1158, 2000.

Williams, D. On optimal auv track-spacing for underwater mine detection. In *IEEE T. Robotic. Autom. 2010*, pages 4755–4762. IEEE, May 2010.

Winston, W. L. and J. B. Goldberg. *Operations research: applications and algorithms*. Duxbury press Belmont, CA, 1987.

Wong, S. C. and B. A. MacDonald. A topological coverage algorithm for mobile robots. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, pages 1685–1690. IEEE, 2003.

Wong, S. C. and B. A. MacDonald. Complete coverage by mobile robots using slice decomposition based on natural landmarks. In *PRICAI 2004: Trends in Artificial Intelligence*, pages 683–692. Springer, 2004.

Xiao, J., Z. Michalewicz, L. Zhang, and K. Trojanowski. Adaptive evolutionary planner/navigator for mobile robots. *IEEE T. Evolut. Comput.*, 1(1):18–28, Apr 1997.

Xu, A., C. Viriyasuthee, and I. Rekleitis. Optimal complete terrain coverage using an unmanned aerial vehicle. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2513–2519. IEEE, 2011.

Yan, Z., Y. Zhao, T. Chen, and L. Jiang. Fault recovery based mission scheduling of auv for oceanographic survey. In *World Congr. Intelligent Control Autom. (WCICA) 2012*, pages 4071–4076, July 2012.

Yang, S. X. and C. Luo. A neural network approach to complete coverage path planning. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(1):718–724, 2004.

Yang, S. X. and M. Meng. An efficient neural network approach to dynamic robot motion planning. *Neural Networks*, 13(2):143–148, 2000.

Yoerger, D. R., A. M. Bradley, B. Walden, M. Cormier, and W. Ryan. High resolution mapping of a fast spreading mid ocean ridge with the autonomous benthic explorer. In *International Symposium on Unmanned Untethered Submersible Technology*, pages 21–31. University of New Hampshire - Marine systems, 1999.

Yu, Y. and Z. H. Zhou. On the usefulness of infeasible solutions in evolutionary search: A theoretical study. In *Proceedings of the IEEE Congress on Evol. Comput.*, pages 835–840. IEEE, 2008.

Zhu, D., W. Li, M. Yan, and S. X. Yang. The path planning of auv based on ds information fusion map building and bio-inspired neural network in unknown dynamic environment. *International Journal of Advanced Robotic Systems*, 11, 2014.

Zitzler, E., M. Laumanns, and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm. Technical report, Swiss Federal Institute of Technology, 2001.