Case-based replanning of search missions using AUVs

Nuno Abreu[1 2] and Aníbal Matos[1 2]
[1]INESC TEC, Campus da FEUP, Rua Dr. Roberto Frias, 378, 4200-465 Porto, Portugal
[2]FEUP - DEEC, Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal
Emails: {nabreu, anibal}@inesctec.pt

***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. The objective of the paper 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. 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 (CBR) in which old problem and solution information helps solve a new problem. The resulting combination takes advantage of both paradigms where our evolutionary approach in conjunction with an artificial neural network (ANN), presented earlier, delivers robustness and adaptive learning while the case-based component speeds up the replanning process. 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.**

I. Introduction

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. 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.

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. 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. 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 fulfills the objective in a timely manner based on acquired sensor data without hogging system resources such as CPU cycles or memory.

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.

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.

II. Contributions

The key contributions of this work include:

- 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 [1]. 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.
- Experiments which demonstrate that the proposed methods are successful and effective in planning search missions.

III. Problem statement

The principal problem under study in this paper 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. The mission is generally considered complete when a proportion of cells have been covered to a specified level C_{thresh}.

IV. Related Work

Coverage strategies can be classified according to several criteria. One important 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 [2]. A comprehensive survey of the existing coverage path planning methods may be found in Galceran et al. [3]. Seto [4] performed a review of the state-of-art of autonomy for underwater vehicles.

A lawn-mowing search pattern [5] with several parallel tracks is standard if no prior information on potential target locations is available. As the area boundary becomes more complex, decomposition methods [6] are used to divide the area into simple subareas.

Hert et al. [7] presented an 3D online CPP algorithm for an AUV moving in an unknown underwater environment. 3D coverage is achieved by applying a 2D terrain-covering algorithm in successive horizontal planes laying at different depths. 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.

Oksanen and Visala [8] developed two (greedy) approaches for solving the 2D CPP in the case of agricultural applications with non omnidirectional vehicles. Their online algorithm, although incremental, plans paths on the basis of the machine's current state and the search is on the next swath instead of the next subfield (the result from area decomposition procedure to identify simpler areas to operate). There are advantages and disadvantages with the algorithm and it does not solve the coverage path planning problem optimally.

Lee et al. [9] presented an online complete CPP and control algorithm for a mobile robot. It focuses on smoothing the coverage path for more energy efficient coverage. However, using a high resolution map requires high processing power which may be a problem for low power embedded computers.

Xu et al. [10] extended an optimal coverage algorithm [11] for the general class of non-holonomic robots, particularly UAVs. 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 online motion planner can adjust the coverage footprint width to compensate for the deviations in trajectory.

Paull et al. [12] present an approach to 2D seabed coverage for minehunting missions using a sidescan sonar. Paths are planned using multi-objective optimization involving information theory and branch entropy based on a hexagonal cell decomposition. One disadvantage of the method is the need to tune the weights in the objective function and the lack of guidance on how to do so.

Candeloro et al. [13] proposed an extension to Paull's [14] planner, aiming to plan coverage missions for finding objects of interest using an underwater vehicle equipped with a camera. It is assumed the vehicle navigates in an unstructured and often poorly known environment but the sea bottom is considered to be flat.

Since many information is available a priori, an optimal 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 processed for online planning can make the search task infeasible if relying on the limited onboard processing power. Thus, there should be a realistic balance between the reduction of the path search space and search performance.

A. Detection performance

In order to assess the effectiveness of a search operation we need to be able to estimate the detection performance that should be achieved in a specific mission.

1) Lateral range: The concept of lateral range curve (LRC) was introduced by Koopman [15]. 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. 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.

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 we are assuming that the object is in the area searched. It has become the de facto measurement used in search and rescue theory. 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.

V. Online mission planning

In this section, the application of the previously presented EA [1] 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.

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. For this reason, a system that uses a case-base as a long term knowledge store in a new planning algorithm is proposed. Our approach borrows ideas from case-based reasoning (CBR) in which old problem and solution information, stored as cases in a case-base, helps solve a new problem. 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 [1], 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 in order to achieve a given level of performance. The casebase stores the best individuals, the Pareto set, found using the offline evolutionary planner. Hence, each case describes 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. Figure 1 represents a flowchart of the online mission planning process.

A. 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. Figure 2(b) 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.

Miller and Campbell [16] 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.

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 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.

B. Clustering

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.

We implemented an agglomerative clustering algorithm, considering the need to agglomerate cells and the existence of a large number of identified areas.

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. 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:

$$d_{min}(C_i, C_j) = \min_{p_i \in C_i, p_j in C_j} \parallel p_i - p_j \parallel \quad (1)$$

It's called "single link" because it says clusters are close if they have even a single pair of close points, a single "link".

In summary, our strategy consists of 3 phases that can be visualized in figure 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.

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). This process is described by algorithm 1.

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.

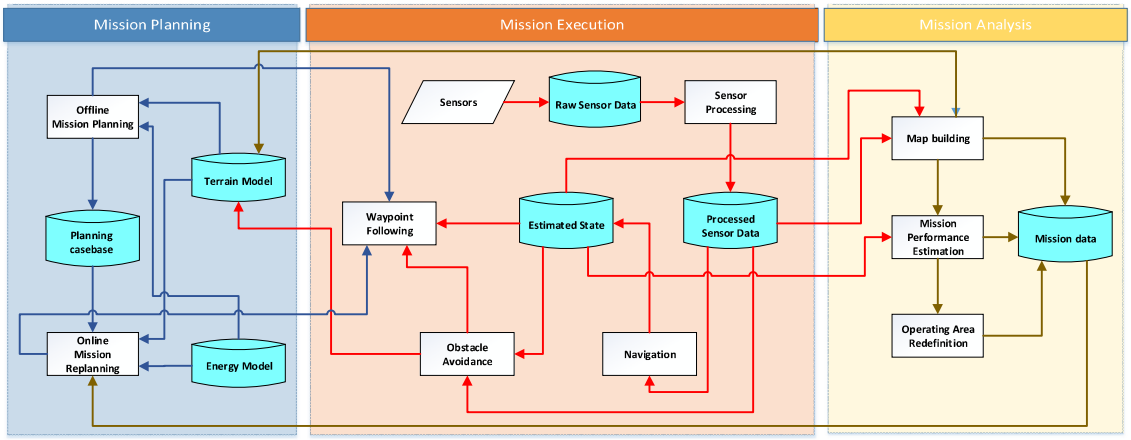
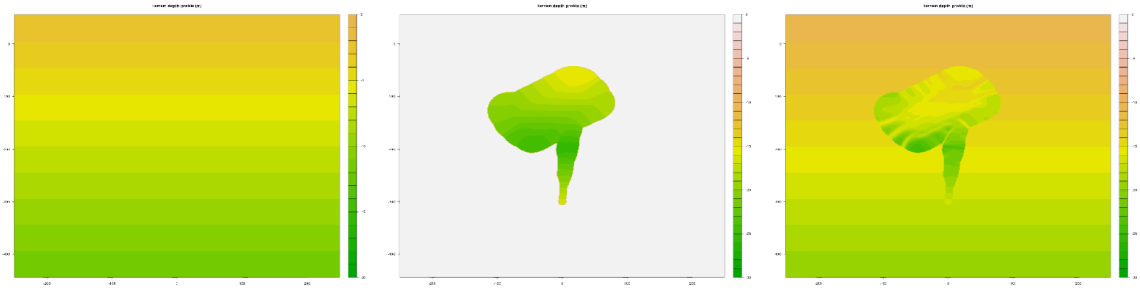


Fig. 1. Mission flowchart.

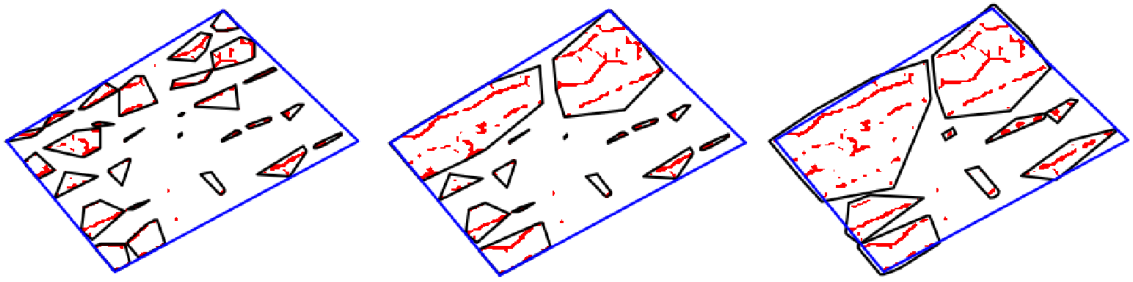


(a) Original DEM.

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

(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.

Fig. 2. Graphical demonstration of the map building algorithm.



(a) Phase 1.

(b) Phase 2.

(c) Phase 3.

Fig. 3. Graphical demonstration of the clustering algorithm.

Algorithm 1: First Phase Clustering Algorithm.

```
function FirstClusteringPhase (grid, settings_minsimilarity);
input : grid(i, j), i = 1, ..., n&j =
        1, ..., m; settings_minsimilarity
output: clusters = {clusters_0, ..., clusters_p}
while size(clusters) > 1 do
    /* Find most similar clusters                    */
    for i ← 1 to length(similaritymatrix) do
        for j ← i+1 to length(similaritymatrix_i) do
            sim = similaritymatrix_ij;
            c_1 = i;
            c_2 = j;
        end
    end
    /* If clusters are not close enough then
       terminate and return clusters                 */
    if sim < settings_minsimilarity then
        return clusters;
    end
    /* merge most similar clusters                   */
    clusters.add(Merge(clusters_c1,clusters_c2));
    clusters.remove(c_1);
    clusters.remove(c_2);
    /* update centroids                              */
    centroids.remove(c_1);
    centroids.remove(c_2);
    centroids.add(GetCentroid(clusters.last));
    //* update similarity                           */
    similaritymatrix.removecolumn(c_1);
    similaritymatrix.removecolumn(c_2);
    similaritymatrix.removerow(c_1);
    similaritymatrix.removerow(c_2);
    for i ← 1 to length(similaritymatrix) do
        similaritymatrix_i.add(
        ClusterSimilarityCentroidDistance(centroids_i,
          centroids.last));
    end
end
return clusters;
```

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 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. This process is described by algorithm 2.

3) Area boundary definition: Finally, once the clusters are determined, the areas that will be used in the replanning process can be defined. 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 is enlarged by the average nadir width for that section, allowing coverage from outside the original area.

Algorithm 2: Second Phase Clustering Algorithm.

```
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                                 */
while size(clusters) >1 do
    /* Find most similar clusters                    */
    for i ← 1 to length(similaritymatrix) do
        for j ← i+1 to length(similaritymatrix_i) do
            sim = similaritymatrix_ij;
            c_1 = i;
            c_2 = j;
        end
    end
    /* If clusters are not close enough then
       terminate and return clusters                 */
    if sim < settings_minsimilarity then
        return clusters;
    end
    /* Test cluster candidate                        */
    clustercandidate = Merge(clusters_c1,clusters_c2);
    hull = ConvexHull(clustercandidate);
    /* area occupied by the cluster                  */
    clusteroccupiedarea = size(clustercandidate)*cellsize;
    hulloccupiedarea = CalculatePolygonArea(hull);
    oversizeratio = hulloccupiedarea/clusteroccupiedarea;
    if oversizeratio < settings_hullmaxratio then
        /* accept new cluster                        */
        clusters.add(clustercandidate);
        clusters.remove(c_1);
        clusters.remove(c_2);
        /* update similarity                         */
        similaritymatrix.removecolumn(c_1);
        similaritymatrix.removecolumn(c_2);
        similaritymatrix.removerow(c_1);
        similaritymatrix.removerow(c_2);
        for i ← 1 to length(similaritymatrix) do
            similaritymatrix_i.add(
            ClusterSimilaritySingleLink(clusters_i,
              clusters.last));
        end
    end
end
return clusters;
```

4) Clustering evaluation: The quality of the created clusters is evaluated using the silhouette coefficient, which is a cluster validity measure.

$$cl_{sil}(i) = \frac{cl_{coh} - cl_{sep}}{\max(cl_{coh}, cl_{sep})}, -1 \leq cl_{sil} \leq 1 \qquad (2)$$

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 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.

C. CBR replanning

Consider the following definitions:

$$A = \{a_1, a_2, ..., a_n\} \tag{3}$$

$$S^i = \{s_1, s_2, ..., s_{n_i}\}, i = 1, ..., n \tag{4}$$

$$A' = \{a'_1, a'_2, ..., a'_m\}, A' \in A \tag{5}$$

$$S'^j = \{s'_1, s'_2, ..., s'_{n_j}\}, j = 1, ..., m \tag{6}$$

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.

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. 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. 4. Given a new task to cover the identified areas, it can choose between planning using a path from the casebase, 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

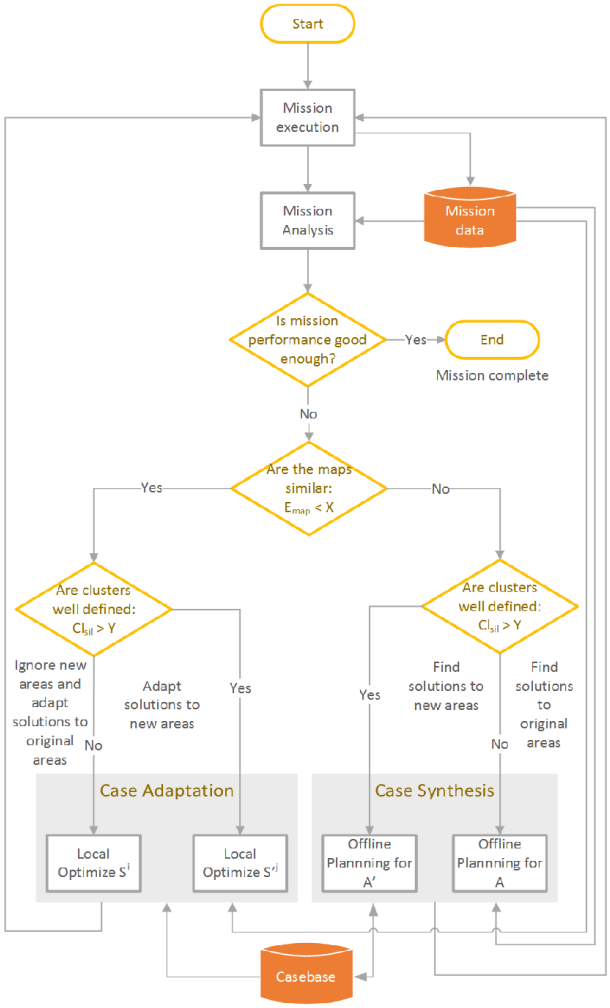


Fig. 4. Mission replanning flowchart.

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.

1) Case Retrieval: The performance of the cases in the casebase needs to be reevaluated according to the new information:

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

2) Case Adaptation: Unless no area was previously covered, then there is always some sort of adaptation performed to the old paths:

- 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;
- perform local optimization of every path of every area a_i';
- update detection performance, energy consumption and mission time duration estimates;

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.

In the same belief that similar problems could have similar solutions [17], it is worthwhile to reuse previous ANN structure to deal with similarly structured environments, instead of performing an exhaustive search for the best topology.

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 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.

To handle this problem the density of individuals in decision space is controlled. 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.

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 which was developed for offline global planning [1].

6) Execution Time Constraints: Guarantee that algorithm is executed when 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.

VI. Experiments

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 DEM contains a grid with 100 meters resolution constructed with depths from the latest hydrographic surveys made on the area. Figure 5 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.

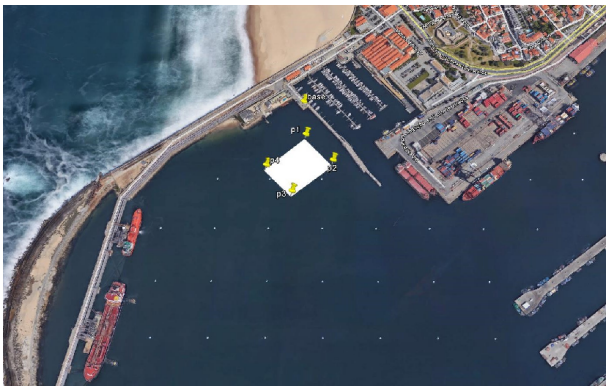
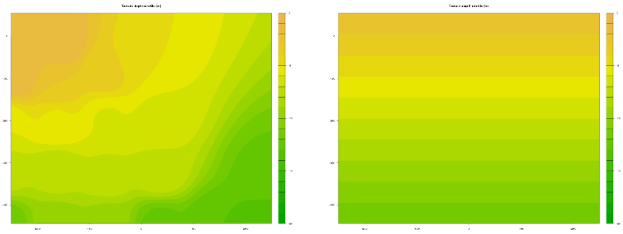


Fig. 5. Google Earth satellite view of the Leixões harbor in Portugal. The operation area is represented by a polygon in white. The blue dots are the measurements from the geospatial database.

Three 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 tests 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;
- 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. The planner will use simplified versions of the DEM 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 which fused heading from the compass, absolute position from the GPS, depth estimated from pressure readings and velocity over ground measurements from the DVL. The performance



(a) Real terrain. (b) Sloped terrain used for offline planning.

Fig. 6. Terrains used for representing the topography in the Leixões harbor.

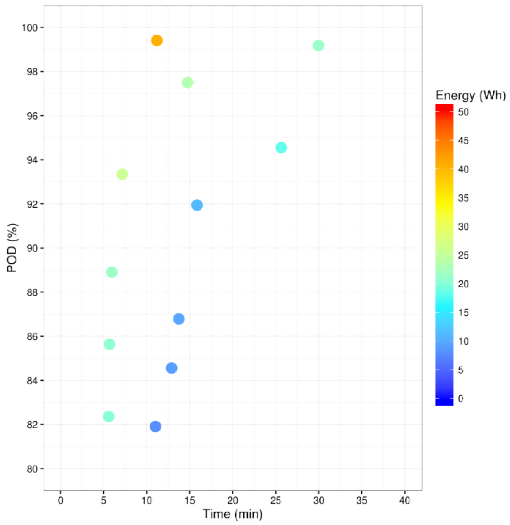
for each test case is established by controlling the amount and quality of data sent to the navigation system. Bad navigation performance is achieved by sending acceleration and angular velocity measurements with high noise and a drifting bias to the INS, without a velocity fix. Average navigation performance 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. 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.

1) Offline planning: The offline planner executed the evolutionary planner, with terrain represented in figure 6(b), for 109 seconds terminating on the fifteenth iteration. As displayed in figure 7(a), 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 7(b) 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.

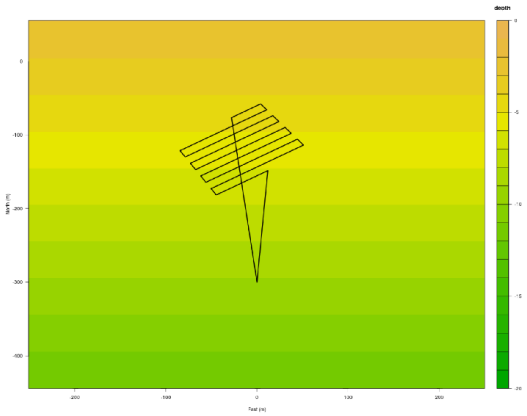
2) Test case A: The planned and real paths followed by the vehicle during test A can be observed in figure 8. As expected, the INS performance without velocity fix was pretty bad and the vehicle was unable to follow the planned trajectory. With a silhouette coefficient of 0.86, the clusters were well defined, as can be seen in figure 9(a). 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 9(a) 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 9(b). That divergence at the end (top left of the figure 8) 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.

3) Test case B: The planned and real paths followed by the vehicle during test B can be observed in figure 10. 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 11(a). 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 11(a) 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 11(b).

4) Test case C: The planned and real paths followed by the vehicle during test C can be observed in figure 12. For our surprise, the performance of the INS, even with 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 of 1.00, the identified cluster actually covered all of the the operation area, as can be seen in figure 13(a). 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 13(a) in black, which coincidentally equal to the original area. Unfortunately, the chosen strategy was unable to generate a good solution to the problem and



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



(b) Solution chosen by offline planning.

Fig. 7. Solutions generated by the evolutionary planner.

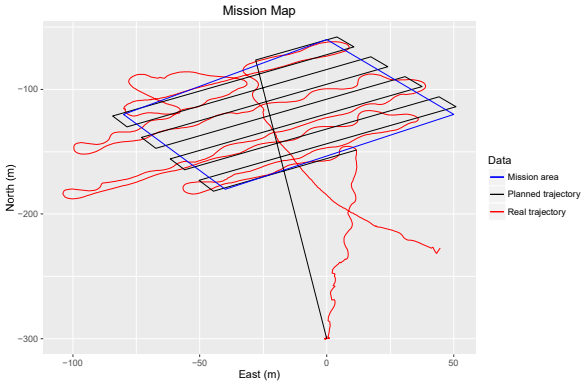
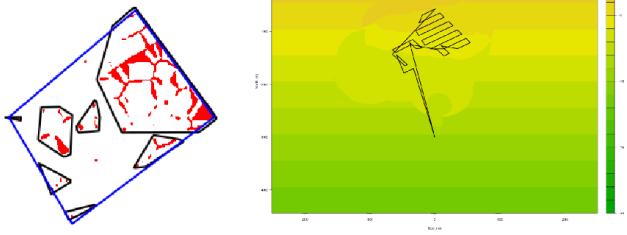


Fig. 8. Plot of the trajectory generated by the offline planner and the real trajectory followed by the vehicle, for test case A.



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

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

Fig. 9. Output of the online planner for test case A.

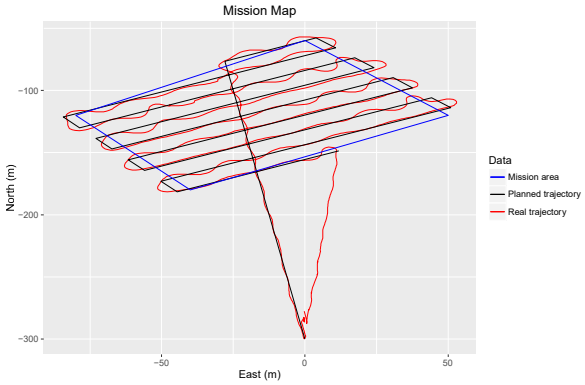
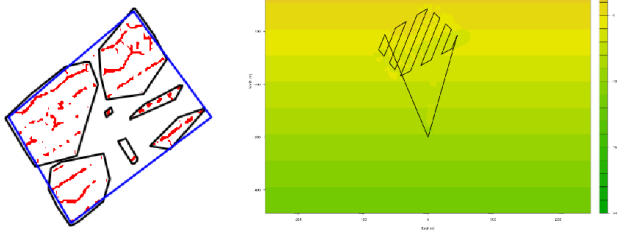


Fig. 10. Plot of the trajectory generated by the offline planner and the real trajectory followed by the vehicle, for test case B.



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

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

Fig. 11. Output of the online planner for test case B.

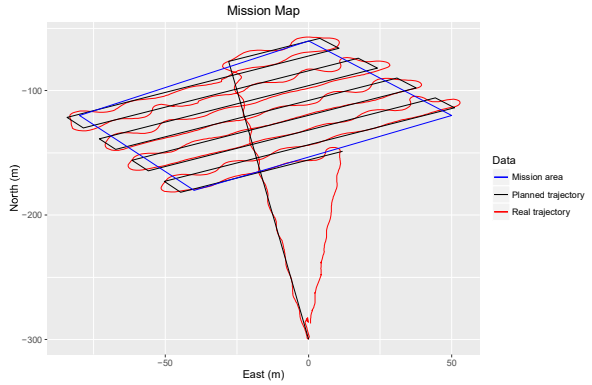
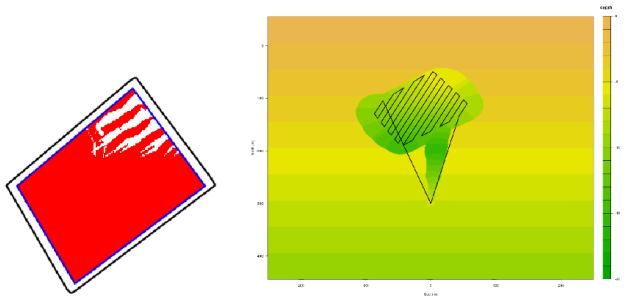


Fig. 12. Plot of the trajectory generated by the offline planner and the real trajectory followed by the vehicle, for test case C.



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

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

Fig. 13. Output of the online planner for test case C.

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 13(b).

VII. Conclusions

This paper extended our multi-objective multi-stage approach for search operations mission planning, combining a EA with simulated annealing and a ANN, in static 3D environment with partially unknown terrain. A novel mission replanning algorithm was presented which reused 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.

Acknowledgment

This work is financed by the ERDF – European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project «POCI-01-0145-FEDER-006961», and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) as part of project UID/EEA/50014/2013.

References

[1] N. Abreu and A. Matos, "Minehunting mission planning for autonomous underwater systems using evolutionary algorithms," *Unmanned Systems*, vol. 2, no. 04, pp. 323–349, 2014.

[2] H. Choset, "Coverage for robotics - a survey of recent results," in *Ann. Math. Artif. Intell.*, SCITEPRESS, 2001.

[3] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[4] M. L. Seto, *Marine robot autonomy*. Springer Science & Business Media, 2012.

[5] S. Land and H. Choset, "Coverage path planning for landmine location," in *3rd Int. Symp. Technol. and the Mine Problem*, 1998.

[6] W. H. Huang, "Optimal line-sweep-based decompositions for coverage algorithms," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 1, pp. 27–32, IEEE, 2001.

[7] S. Hert, S. Tiwari, and V. Lumelsky, "A terrain-covering algorithm for an auv," in *Underwater Robots*, pp. 17–45, Springer, 1996.

[8] T. Oksanen and A. Visala, "Coverage path planning algorithms for agricultural field machines," *Journal of Field Robotics*, vol. 26, no. 8, pp. 651–668, 2009.

[9] T.-K. Lee, 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*, vol. 59, no. 10, pp. 801–812, 2011.

[10] A. Xu, C. Viriyasuthee, and I. Rekleitis, "Optimal complete terrain coverage using an unmanned aerial vehicle," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 2513–2519, IEEE, 2011.

[11] R. Mannadiar and I. Rekleitis, "Optimal coverage of a known arbitrary environment," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 5525–5530, IEEE, 2010.

[12] L. Paull, S. Saeedi, M. Seto, and H. Li, "Sensor-driven online coverage planning for autonomous underwater vehicles," *Mechatronics, IEEE/ASME Transactions on*, vol. 18, no. 6, pp. 1827–1838, 2013.

[13] M. Candeloro, F. Mosciaro, A. J.Sørensen, G. Ippoliti, and M. Ludvigsen, "Sensor-based autonomous path-planner for sea-bottom exploration and mosaicking," *IFAC-PapersOnLine*, vol. 48, no. 16, pp. 31 – 36, 2015.

[14] L. Paull, S. Saeedi, M. Seto, and H. Li, "Sensor driven online coverage planning for autonomous underwater vehicles," in *IEEE Int. Conf. Int. Robot (IROS) 2012*, pp. 2875–2880, Oct 2012.

[15] B. Koopman, *Search and Screening: General Principles with Historical Applications*. Virginia, USA: The Military Operations Research Society, 1999.

[16] I. Miller and M. Campbell, "A mixture-model based algorithm for real-time terrain estimation," in *The 2005 DARPA Grand Challenge*, pp. 407–436, Springer, 2007.

[17] D. B. Leake and D. C. Wilson, "When experience is wrong: Examining cbr for changing tasks and environments," in *Case-Based Reasoning Research and Development*, pp. 218–232, Springer, 1999.