# Path Planning with Homotopy Class Constraints on Bathymetric Maps

Emili Hernández and Marc Carreras and Enric Galceran and Pere Ridao
Department of Computer Engineering
University of Girona, 17071 Girona, Spain
Email:{emilihb,marcc,enricgalceran,pere}@eia.udg.edu

***Abstract*—This paper is presented in the context of the TRIDENT European project. The paper proposes a path planning approach that uses homotopy classes to guide the path search in bathymetric maps. First, it turns the bathymetry into a 2D workspace. Then, it builds a topological environment based on the workspace to compute homotopy classes, which topologically describe how paths go through the obstacles in the workspace. The homotopy classes are sorted according to an heuristic estimation of their lower bound. Then, a path planner based on the A*, called *Homotopic A** (HA*), guides the path search in the workspace using the generated homotopy classes. Simulated and real results with a bathymetric map obtained with a Multibeam Profiling Sonar (MPS) sensor in the Catalan coast show the feasibility of the proposal.**

## I. Introduction

The work presented in this paper is part of the TRIDENT European project [1], which proposes a new methodology for multipurpose underwater intervention tasks with diverse potential applications like underwater archaeology, oceanography and offshore industries, going beyond present-day methods typically based on manned and/or purpose built systems. A team of two cooperative heterogeneous robots with complementary skills, an Autonomous Surface Craft (ASC) and an Intervention Autonomous Underwater Vehicle (I-AUV) endowed with a dexterous manipulator, will be used to perform underwater manipulation tasks.

The experiments of the project consist of two steps. In the first step, the I-AUV is deployed from the ASC to perform a path following survey, where it gathers optical/acoustic data from the seafloor to do an accurate terrain tracking. After the survey, the I-AUV docks with the ASC and sends the data back to a ground station where a map is set up and a target object is identified by the end user. At the second step, the ASC navigates towards a waypoint near the intervention area where the I-AUV is launched to search for the object. When the object has been found, the I-AUV switches to free floating navigation mode to start the manipulation process.

The I-AUV that will be used to carry out the experiments is the Girona 500 [2] (Fig. 1). It is equipped with a phased array DVL Explorer from RDI, an Attitude and Heading Reference System (AHRS) from Tritech composed of an Intelligent Gyro Compass (IGC) for attitude and an Intelligent Fiber-optic Gyro for heading, a Linquest USBL 1500HA with modem, a Super Seaking dual frequency profiling sonar form Tritech, a Sound Velocity System (SVS) with a pressure sensor form Valeport,

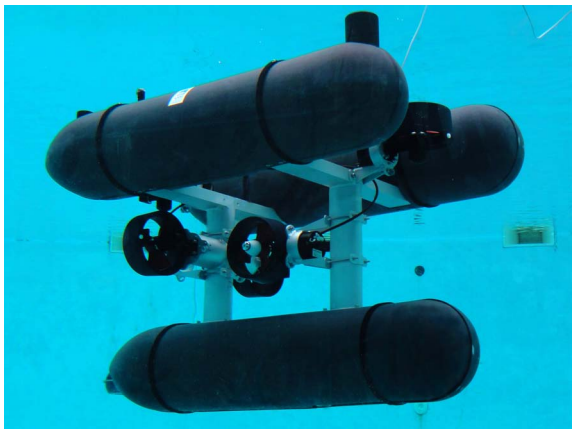


Fig. 1. Girona 500 I-AUV

a GPS sensor, an Imaginex Sidescan sonar, a Tritech SeaSpy CCD camera and an Imagenex Multibeam Profiling Sonar (MPS).

This paper presents a path planning approach to be used in the second step of the experiments, when the I-AUV has to compute safe paths for the intervention based on the generated map. Our approach is based in the work presented in [3], where we generate homotopy classes that can be followed in any 2D workspace. Homotopy classes describe topologically how paths avoid obstacles while constraining the zones of the workspace where a path can go through. Therefore, a user can choose specific homotopy classes to follow according a set of criteria, discarding those whose paths that, for instance, go through zones affected by strong currents or are potentially dangerous. Then, we apply a search-based algorithm based on the A* algorithm, called *Homotopic A** (HA*) [4], which computes the optimal path in the workspace for a given homotopy class.

In this paper, we extend our work by applying a lower bound estimator to the homotopy classes that allows to know those classes that most-probably contain the lower cost solution before computing the path in the workspace. Thus, the algorithm can generate some good solutions very fast. The reliability and performance of our method has been tested in simulation. Moreover, we also present the results of a preliminary experiment carried out with the MPS mounted on a boat where DGPS positioning was also available (Fig. 2). Using the

978-1-61284-4577-0088-0/11/$26.00 ©2011 IEEE

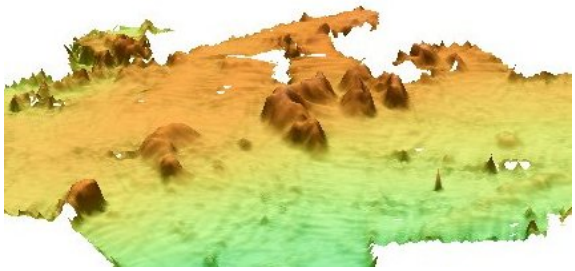
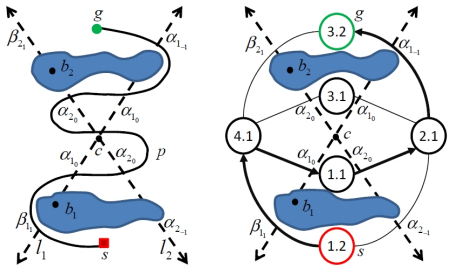


Fig. 2. Bathymetric map generated with a commercial software.




a) Reference frame b) Topological graph

Fig. 3. Topological path represented in the reference frame as $p = \beta_{1_1}\alpha_{1_0}\alpha_{2_0}\alpha_{1_0}\alpha_{2_0}\alpha_{2_0}\alpha_{1_0}\alpha_{1_{-1}}$ and its canonical sequence $(\beta_{1_1}\alpha_{1_0}\alpha_{2_0}\alpha_{1_{-1}})$ in the topological graph.

bathymetric map and considering the depth where the vehicle was intended to operate for achieving the intervention position, a 2D Occupancy Grid Map (OGM) was generated to perform our method.

The paper is structured as follows. Section II describes the method to obtain homotopy classes from a metric environment. Section III describes the topological-guided path planning algorithm to compute paths in the workspace that accomplish homotopy class constraints. Section IV reports the results, and section V exposes the conclusions and future work.

## II. Homotopy Classes of the Workspace

Given a workspace with obstacles, in [3] we propose a method to generate a topological representation of the environment which is used to compute all the different homotopy classes from the start point to the end point. Two paths belong to the same homotopy class if one can be deformed into the other one without encroaching any obstacle.

### A. Reference Frame

The reference frame determines, in the metric space, the topological relationships between obstacles and it is used to name the homotopy classes. The whole construction process is summarized in three steps:

1) Select a random point inside each obstacle and label it as $b_k$, where $k = 1..n$.
2) Select the central point $c$ of the reference frame. This point cannot be inside an obstacle nor being inside the $n(n-1)/2$ lines determined by the pairwise choices of distinct $b_k$.
3) Construct $n$ lines $l_k$ joining $c$ with each $b_k$. Each line is partitioned into $m+1$ segments, where $m$ is the number of obstacles that intersect with $l_k$ in the workspace. The segments from $b_k$ and away from $c$ are labeled with $\beta_{k_s}$, and the segments in the opposite direction are labeled $\alpha_{k_s}$, where $s = 0..u$ with $u \in \mathbb{Z}^+$ for the segments of $l_k$ from $c$ that passes through $b_k$ and $s = 0..v$ with $v \in \mathbb{Z}^-$ for the segments in the opposite direction.

Using the reference frame, any path $p$ can be defined by the sequence of labels of the segments being crossed in order from the starting to the ending point. For instance, Fig. 3a depicts a reference frame for a scenario with two obstacles. The path that traverses it is labeled $\beta_{1_1}\alpha_{1_0}\alpha_{2_0}\alpha_{1_0}\alpha_{2_0}\alpha_{2_0}\alpha_{1_0}\alpha_{1_{-1}}$. There are two special cases: when $p$ crosses no rays then $p = \emptyset$ and when $p$ crosses through $c$ meaning that all the $\alpha$'s are simultaneously crossed. In such a latter case, all $\alpha_{k_s}$ are added in subindex order to the sequence.

Two paths are homotopic if they have the same *canonical sequence*, which is the simplest representation of a path without changing its topology. With the notation used, it is computed by sorting the $\alpha$'s substrings of the path in non-decreasing order of subindex and then removing all the elements of the sequence by pairs that have the same character. This process is repeated until no changes are made to the sequence. In Fig. 3 the canonical sequence of the path $\beta_{1_1}\alpha_{1_0}\alpha_{2_0}\alpha_{1_0}\alpha_{2_0}\alpha_{2_0}\alpha_{1_0}\alpha_{1_{-1}}$ is $\beta_{1_1}\alpha_{1_0}\alpha_{2_0}\alpha_{1_{-1}}$.

### B. Topological Graph

The topological graph $G$, whose construction is based on the reference frame, provides a model to describe the topological relationships between regions of the metric space. Its construction can be divided in three steps:

1) The lines of the reference frame divide the metric space into regions or *wedges* and the obstacles that intersect with more than one line at the same time split these wedges into *sub-wedges*. Each sub-wedge represents a node of $G$.
2) Each node of $G$ is labeled according to the wedge $w$ and sub-wedge $sw$ using the notation $w.sw$. $w \in \mathbb{N}$ is numbered counterclockwise. For each $w$, its corresponding $sw \in \mathbb{N}$ are numbered sequentially starting by 1 for the one closest to $c$.
3) Two nodes of $G$ are interconnected according to the number of segments they share in the reference frame. Each edge of $G$ is labeled with the same label of the segment that crosses in the reference frame.

In the reference frame, a path is defined according to the segments it crosses whereas in $G$ it turns into traversing the graph from the starting node to the ending node[1]. Fig. 3a depicts a path in the reference frame and Fig. 3b its equivalent description in the topological graph.

[1]Starting and ending nodes are those *wedges* in the reference frame -nodes in $G$- where the starting and ending points are located.

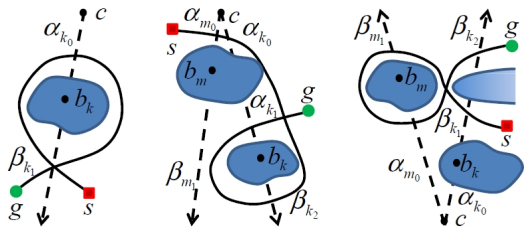



Fig. 4. Examples of the restriction criteria.

### C. Generation of Homotopy Classes

Once the topological graph is constructed, it is traversed using a modified version of the BFS algorithm used in [3]. The BFS is a graph search algorithm that begins at the root node and explores all the neighboring nodes. Then, for each of those nearest nodes, it explores their unexplored neighbors. The process is repeated until the goal is found. Unlike the standard BFS, which stops when all vertexes have been visited, the algorithm continues until there are no more homotopy class candidates to explore or the length of the last homotopy class candidate is larger than a given threshold.

During the BFS execution, several restriction criteria are applied to avoid the generation of any homotopy class which either self-intersects or whose canonical sequence is duplicated and has already been considered. All classes that accomplish any of the following restrictions criteria are ignored to avoid using them as a root for future homotopy classes:

- Simple wrap. Any string that contains a substring of the form $\alpha_{k_s}...\chi_{k_t}...\alpha_{k_u}$ or $\beta_{k_s}...\chi_{k_t}...\beta_{k_u}$ where $\chi = (\alpha, \beta)$ with $s = u$ represents a class that wraps around an obstacle and is self-crossing.
- Wrap. Any string that contains a substring of the form $\chi_{k_s}...\chi_{k_t}...\chi_{k_u}$ where $\chi = (\alpha, \beta)$ with $s, t, u \geq 0$ and $s > t < u$ or with $s, t, u \leq 0$ and $s < t > u$ represents a class that wraps around an obstacle and is self-crossing.
- Self-crossing. Any string that contains a substring of the form $\chi_{k_s}...\beta_{m_t}...\alpha_{m_u}...\chi_{k_v}$ where $\chi = (\alpha, \beta)$ with $s, v \geq 0$ and $s < v$ or with $s, v \leq 0$ and $s > v$ represents a class that self-crosses. The reversed substring $\chi_{k_s}...\alpha_{m_t}...\beta_{m_u}...\chi_{k_v}$ with $s, v \geq 0$ and $s > v$ or with $s, v \leq 0$ and $s < v$ also represents a class that self-crosses.

  Fig. 4a depicts an example of the simple wrap criterion with path $\beta_{k_1}\alpha_{k_0}\beta_{k_1}$. Fig. 4b shows a wrap with the path $\alpha_{m_0}\alpha_{k_0}\beta_{k_2}\alpha_{k_1}$ and Fig. 4c depicts an example of the self-crossing criterion with path $\beta_{k_1}\beta_{m_1}\alpha_{m_0}\beta_{k_2}$.
- Duplicated strings are not allowed in the list of homotopy class candidates. If a string is not in its canonical form, it can be simplified without modifying its topology. Then, it is ensured that the resultant string has been already computed by the BFS algorithm because it would be shorter than the input string. Finally, the algorithm cannot traverse through the same edge on two consecutive occasions. By doing that, a string with a repeated pair would be generated. Consequently, the pair would be simplified and the string discarded for being duplicated.

### D. Lower Bound

Depending on the number of homotopy classes generated by the BFS algorithm, it is not possible to compute all their correspondent paths in the workspace in real-time. Therefore, we have modified *the funnel algorithm* [5] to obtain a quantitative measure for each homotopy class estimating their quality. This algorithm computes the shortest path within a $channel$, which is a polygon formed by the vertexes of the segments of the reference frame that are traversed in the topological graph. The modification consists in accumulating the Euclidean distance between the points while they are being added to the shortest path. Hence, the result of the funnel algorithm is a lower bound of the optimal path in the workspace of the selected homotopy class. It is used to set up a preference order to compute the homotopy classes path in the workspace when operating under time restrictions. Notice that the segments of the reference frame constrain the region where the paths can go through, but do not take into account the shape of the obstacles. For that reason, an homotopy class with a smaller lower bound may have a longer path in the workspace than an another homotopy class with a higher lower bound.

## III. Guided path planning

Once the homotopy classes are computed, a path planning algorithm has to find a path in the workspace that follows a given homotopy class, which essentially means to turn a topological path into a metric one. The only link between the workspace and the topological space is the reference frame. It allows checking whether a metric path in the workspace is following a topological path by checking the intersections –in order– from the initial configuration to the current configuration. We propose a variation of the A* algorithm called *Homotopic A** (HA*), which just allows to explore the zones of the workspace that satisfy a given homotopy class

The algorithm is detailed in Alg. 1. The states of the algorithm are tuples that contain the configuration of the robot $q$ and the topological path from $q_{start}$ to $q$. These values are accessible through the functions $Q$ and $P$ respectively. Just like the A*, the visited states are stored in a list $V$ and the open states are processed according their position in a priority queue $OPEN$. Each state in this queue is ordered according to the sum of its current path cost from the start, $g(s)$, and an heuristic estimate of its path cost to the goal, $h(s, s_{goal})$. The state with the minimum sum is at the top of the priority queue.

The algorithm receives as input the start configuration $q_{start}$, the goal configuration $q_{goal}$, a candidate homotopy class to follow $H$ and the reference frame $F$. The configurations $q_{start}$ and $q_{goal}$ are used to set up the initial state $s_{start}$ and the goal state $s_{goal}$ (line 34). The function $ComputePath$ computes the shortest path that follows $H$. It starts by adding the $s_{start}$ into the $OPEN$ queue. While $OPEN$ is not empty, the function pops the state $s$ at the top of the queue. If

$s = s_{goal}$ (line 14), then a path that follows $H$ have been found and the function returns with success. Otherwise, for all the configurations $q'$ reachable from $Q(s)$, the function $FindIntersections$ (line 18) returns the intersections of the segment [$Q(s)$, $q'$] with $F$ sorted by distance[2]. Then the $UpdatePath$ (line 19) generates the new topological path $p$ according to the intersections. No intersection with $F$ means that the explored configuration is in the same subwedge of the workspace and the function returns $P(s)$. If there are intersections and these intersections follow $H$, the function returns $P(s) \cup I$ in order to create a candidate new state $s'$. If $s'$ has already been visited (line 21) and its cost $g(s')$ plus the cost of traversing from $s$ to $s'$, $c(s, s')$ is less than its current cost (line 22), $g(s')$ is set to this new, lower value. If $s'$ does not exists in $V$, it is added into both, the $OPEN$ queue and the visited nodes list $V$.

The completeness of the HA* is ensured because when $s_{goal}$ is not reachable, the algorithm will explore all the states in $OPEN$ before returning that no path has been found. On the other side, when $s_{goal}$ can be reached, the HA* will find the solution following the process described in the paragraph before.

## IV. Results

The method proposed in this paper have been implemented and tested in different scenarios. To identify the obstacles of the scenarios, we have adapted a Component-Labeling algorithm (CL) that efficiently labels connected cells and their contours in greyscale images at the same time [6]. For the construction of the reference frame, the $c$ point has been set at a fixed position in order to ensure the same topological graph construction –and homotopy classes generation– through different executions. The homotopy classes have been set at a maximum of 20 characters length. In order to show all the possible results, no time restrictions have been taken into consideration.

### A. Simulated Results

Our method has been applied in a cluttered environment using a 200x200 pixels bitmap. Fig. 5 depicts the scenario with the paths of the best homotopy classes, according to their lower bound. The construction of the reference frame, the topological graph and the generation of the homotopy classes with their lower bound computation took $7.9ms$. Table I shows the homotopy classes sorted by their lower bound with the path cost and the accumulated computation time, which takes into account the homotopy classes computation and the path generation. The lower bound and the path cost have been normalized with the cost of the optimal path computed with the A* algorithm. The time to build the topological graph, generate the homotopy classes with their lower bound and compute the path with the HA* for the 13 homotopy classes was $4.774s$.

[2]Notice that it is possible to intersect with more than one segment of the reference frame depending on how close $Q(s)$ and $q'$ are to the $c$ point.

**Algorithm 1** Homotopic A*

**FindIntersections**$([q, q'], F)$
1: $r \leftarrow \emptyset$
2: **for** $i \leftarrow 1$ **to** $|F|$ **do**
3: **if** $x \leftarrow Intersection([q, q'], F[i]) \neq null$ **then**
4: $r \leftarrow r \cup \{Edge(i), Distance(q, x)\}$
5: **end if**
6: **end for**
7: $r \leftarrow SortByDistance(r)$
8: **return** $r$

**ComputePath**$(s_{start}, s_{goal}, F)$
9: $OPEN \leftarrow \emptyset$; $V \leftarrow \emptyset$
10: $OPEN.push(s_{start})$
11: **repeat**
12: $s \leftarrow OPEN.top()$
13: $OPEN.pop()$
14: **if** $s = s_{goal}$ **then**
15: **return** $true$
16: **end if**
17: **for all** $q' \in Succ(Q(s))$ **do**
18: $I \leftarrow FindIntersections([Q(s), q'], F)$
19: **if** $p \leftarrow UpdatePath(P(s), I)$ **then**
20: $s' \leftarrow \{q', p\}$
21: **if** $Exists(V(s'))$ **then**
22: **if** $g(s') + c(s, s') < g(V(s'))$ **then**
23: $g(s') \leftarrow g(s) + c(s, s')$
24: **end if**
25: **else**
26: $g(s') \leftarrow g(s') + c(s, s')$
27: $OPEN.push(s')$ with $g(s') + h(s', s_{goal})$
28: $V \leftarrow V \cup s'$
29: **end if**
30: **end if**
31: **end for**
32: **until** $|OPEN| > 0$
33: **return** $false$

**HA***$(q_{start}, q_{goal}, H, F)$
34: $s_{start} \leftarrow \{q_{start}, \emptyset\}$; $s_{goal} \leftarrow \{q_{goal}, H\}$
35: $ComputePath(s_{start}, s_{goal}, F)$

### B. Experimental Results

The method described in this paper has been tested on a bathymetric map gathered with an Imagenex Model 837B "Delta T" 1000 Multibeam Profiling Sonar (MPS). This sensor is a multiple receiver sonar system designed to provide video-like imaging using sonar technology. The MPS has 480 beams spread in a 120° swath overture, the beam rate frequency is between 5-10$Hz$ depending on the depth of the scanned area. The sensor has a motion reference unit sensor to capture roll, pitch and heading.

The experiment took place in the Formigues Islands, in the Catalan Coast. The MPS was fasten to a mast with a DGPS sensor. The mast was attached to a boat to do a survey mission in an area of 100x58$m$. The datasets gathered with the MPS and the DGPS were merged with the commercial software provided by Imagenex to generate the bathymetric map, which is depicted in Fig. 6 with a $0.2m$ resolution.

In the experiment we supposed that a vehicle would navigate $7.5m$ depth. Therefore, using Occupacy Grid Mapping

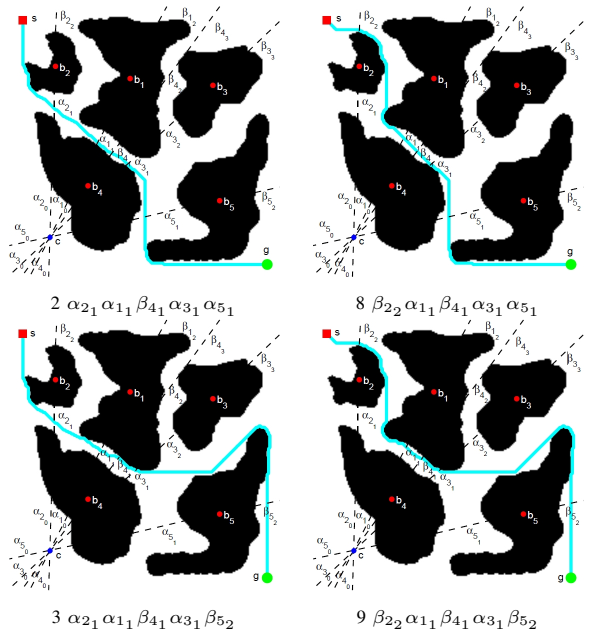


Fig. 5. Paths for the four homotopy classes with the smaller lower bound in Table I.

| Idx | Homotopy class | Lower bound | Cost | Cumulative time (s) |
|---|---|---|---|---|
| 2 | $\alpha_{2_1}\alpha_{1_1}\beta_{4_1}\alpha_{3_1}\alpha_{5_1}$ | 0.83 | 1.00 | 0.236 |
| 8 | $\beta_{2_2}\alpha_{1_1}\beta_{4_1}\alpha_{3_1}\alpha_{5_1}$ | 0.84 | 1.03 | 0.513 |
| 3 | $\alpha_{2_1}\alpha_{1_1}\beta_{4_1}\alpha_{3_1}\beta_{5_2}$ | 0.90 | 1.12 | 0.844 |
| 9 | $\beta_{2_2}\alpha_{1_1}\beta_{4_1}\alpha_{3_1}\beta_{5_2}$ | 0.91 | 1.23 | 1.198 |
| 1 | $\alpha_{5_0}\alpha_{3_0}\alpha_{4_0}\alpha_{1_0}\alpha_{2_0}$ | 0.98 | 1.05 | 1.291 |
| 11 | $\beta_{2_2}\beta_{1_2}\beta_{4_2}\alpha_{3_2}\beta_{5_2}$ | 0.99 | 1.28 | 1.697 |
| 10 | $\beta_{2_2}\beta_{1_2}\beta_{4_2}\alpha_{3_2}\alpha_{5_1}$ | 1.01 | 1.29 | 2.153 |
| 13 | $\beta_{2_2}\beta_{1_2}\beta_{4_3}\beta_{3_3}\beta_{5_2}$ | 1.05 | 1.11 | 2.445 |
| 5 | $\alpha_{2_1}\beta_{1_2}\beta_{4_2}\alpha_{3_2}\beta_{5_2}$ | 1.16 | 1.61 | 2.813 |
| 4 | $\alpha_{2_1}\beta_{1_2}\beta_{4_2}\alpha_{3_2}\alpha_{5_1}$ | 1.18 | 1.62 | 3.219 |
| 12 | $\beta_{2_2}\beta_{1_2}\beta_{4_3}\beta_{3_3}\alpha_{5_1}$ | 1.19 | 1.49 | 3.876 |
| 7 | $\alpha_{2_1}\beta_{1_2}\beta_{4_3}\beta_{3_3}\beta_{5_2}$ | 1.22 | 1.43 | 4.189 |
| 6 | $\alpha_{2_1}\beta_{1_2}\beta_{4_3}\beta_{3_3}\alpha_{5_1}$ | 1.36 | 1.82 | 4.774 |

TABLE I
HOMOTOPY CLASSES OF FIG. 5 ENVIRONMENT SORTED BY THEIR LOWER BOUND.

techniques (OGM) [7], the cells of the bathymetric map with a lower depth were mapped as occupied and the cells with a higher depth were mapped as free. Fig. 7 depicts the resultant 2D workspace as a 500x290 bitmap. The construction of the reference frame, the topological graph and the generation of 45 homotopy classes with their lower bound computation took 0.273$s$. Fig. 8 depicts the normalized cost with respect to the optimal path cost computed with the A* algorithm and the computation time for each homotopy class sorted by their normalized lower bound. Table II shows the five best homotopy classes according to their lower bound and their paths are depicted in Fig. 7.

Changing the start point and/or the goal point the number of homotopy classes generated may change. For instance, Fig. 9 depicts the five best homotopy classes according to their lower bound when a different goal point is selected (Table III). Our method computed 75 homotopy classes in 0.292$s$. The cost for each homotopy class with its lower bound and computation

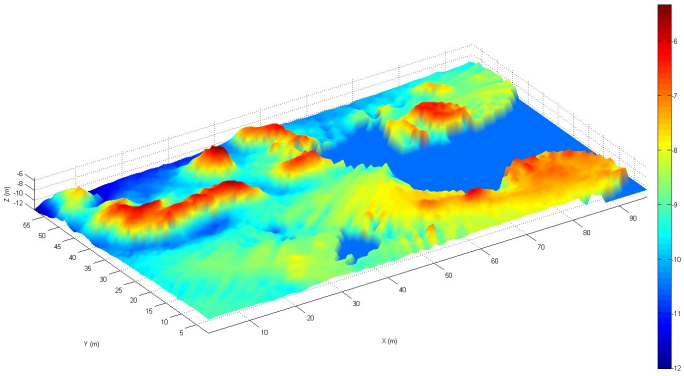


Fig. 6. Bathymetric map obtained in the Formigues Islands.

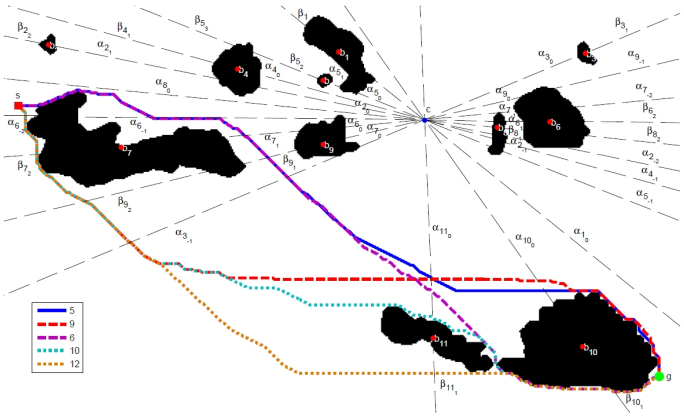


Fig. 7. Paths of the five homotopy classes with the smaller lower bound. The class associated to the index can be found in Table II.

time are shown in Fig. 10.

| Idx | Homotopy class |
|---|---|
| 5 | $\alpha_{6_{-1}}\alpha_{7_1}\beta_{9_1}\alpha_{3_{-1}}\alpha_{11_0}\alpha_{10_0}$ |
| 9 | $\alpha_{6_{-2}}\beta_{7_2}\beta_{9_2}\alpha_{3_{-1}}\alpha_{11_0}\alpha_{10_0}$ |
| 6 | $\alpha_{6_{-1}}\alpha_{7_1}\beta_{9_1}\alpha_{3_{-1}}\alpha_{11_0}\beta_{10_1}$ |
| 10 | $\alpha_{6_{-2}}\beta_{7_2}\beta_{9_2}\alpha_{3_{-1}}\alpha_{11_0}\beta_{10_1}$ |
| 12 | $\alpha_{6_{-2}}\beta_{7_2}\beta_{9_2}\alpha_{3_{-1}}\beta_{11_1}\beta_{10_1}$ |

TABLE II
THE FIVE HOMOTOPY CLASSES WHOSE PATHS HAVE THE LOWER COST WITH THEIR INDEX IN FIG. 7 SCENARIO.

## V. CONCLUSIONS AND FUTURE WORK

This paper presents a path planning approach to be used in the experiments of the TRIDENT European project, where an I-AUV has to compute safe paths towards a goal for intervention purposes on a previously generated map. It is assumed that the I-AUV has to navigate at a fixed depth, hence the vehicle can work with 2D workspaces. Our method is based on the generation of homotopy classes that can be followed in any 2D workspace. Once the bathymetric map has been turned in a 2D map using OGM techniques, we first construct a reference frame which allows representation of any path in the workspace as an homotopy class. Once computed, they are sorted using a lower bound estimator, which classifies

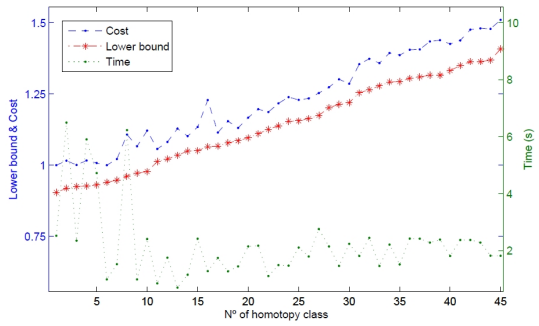



Fig. 8. Normalized cost, normalized lower bound and computation time for paths generated with the HA* for each homotopy class in Fig. 7.

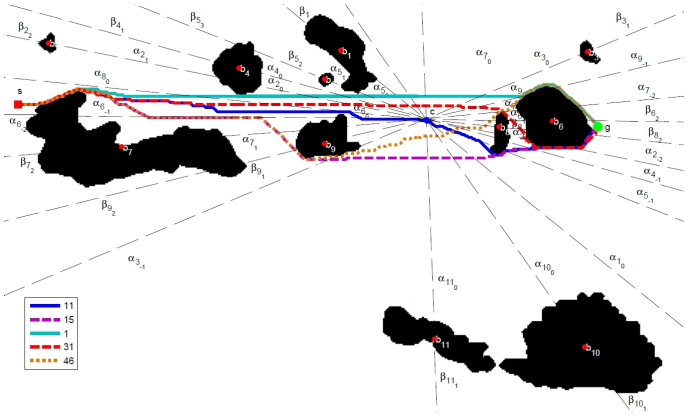



Fig. 9. Paths of the five homotopy classes with the smaller lower bound. The class associated to the index can be found in Table III.

the homotopy classes according to their expected path cost in the workspace. Then, we apply the HA*, a path planning algorithm that computes paths in the workspace following the homotopy classes previously found. The effectiveness of the algorithm has been shown in simulation. Moreover, we also present the results of a preliminary experiment carried out in the Catalan coast with a MPS mounted on a boat where DGPS positioning was also available. Using the resultant bathymetric map we generated the OGM at a fixed depth where our path planning method was applied.

Future work will consist in using the paths generated with our method to guide a vehicle autonomously. We also plan to extend the computation of the homotopy classes in 3D environments to face the major common path planning situations in underwater robotics.

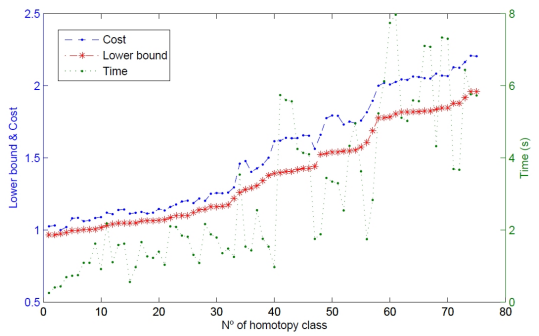



Fig. 10. Normalized cost, normalized lower bound and computation time for paths generated with the HA* for each homotopy class in Fig. 9.

| Idx | Homotopy class |
|---|---|
| 11 | $\alpha_{6_0}\alpha_{7_0}\alpha_{9_0}\alpha_{3_0}\alpha_{11_0}\alpha_{10_0}\alpha_{1_0}\alpha_{5_{-1}}\alpha_{4_{-1}}\alpha_{2_{-2}}\beta_{8_2}$ |
| 15 | $\alpha_{6_{-1}}\alpha_{7_1}\beta_{9_1}\alpha_{3_{-1}}\alpha_{11_0}\alpha_{10_0}\alpha_{1_0}\alpha_{5_{-1}}\alpha_{4_{-1}}\alpha_{2_{-2}}\beta_{8_2}$ |
| 1 | $\alpha_{8_0}\alpha_{2_0}\alpha_{4_0}\alpha_{5_0}\alpha_{1_0}\alpha_{10_0}\alpha_{11_0}\alpha_{3_0}\alpha_{9_{-1}}\alpha_{7_{-2}}\beta_{6_2}$ |
| 31 | $\alpha_{8_0}\alpha_{2_0}\alpha_{4_0}\alpha_{5_0}\alpha_{1_0}\alpha_{10_0}\alpha_{11_0}\alpha_{3_0}\alpha_{9_0}\alpha_{7_{-1}}\alpha_{6_1}\beta_{8_1}\alpha_{2_{-1}}\alpha_{2_{-2}}\beta_{8_2}$ |
| 46 | $\alpha_{6_{-1}}\alpha_{7_1}\beta_{9_1}\alpha_{3_{-1}}\alpha_{11_0}\alpha_{10_0}\alpha_{1_0}\alpha_{5_0}\alpha_{4_0}\alpha_{2_0}\alpha_{8_0}\alpha_{6_0}\alpha_{7_0}\alpha_{9_0}\cdots$ $\alpha_{9_{-1}}\alpha_{7_{-2}}\beta_{6_2}$ |

TABLE III
THE FIVE HOMOTOPY CLASSES WHOSE PATHS HAVE THE LOWER COST WITH THEIR INDEX IN FIG. 9 SCENARIO.

## ACKNOWLEDGMENT

This work was partially supported by the Spanish government under the grant DPI2008-06545-C03-03 and the TRIDENT EU FP7-Project under the grant agreement No: ICT-248497.

## REFERENCES

[1] "Trident website," http://eia.udg.edu/∼dribas, Online April 2011.
[2] D. Ribas, P. Ridao, L. Magí, N. Palomeras, and M. Carreras, "The girona 500, a multipurpose autonomous underwater vehicle," in *Proceedings of the Oceans IEEE*, Santander, Spain, June 2011.
[3] E. Hernández, M. Carreras, J. Antich, P. Ridao, and A.Ortiz, "A topologically guided path planner for an AUV using homotopy classes," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.
[4] E. Hernández, M. Carreras, J. Antich, P. Ridao, and A. Ortiz, "A search-based path planning algorithm with topological constraints. Application to an AUV," in *Proceedings of the 18th IFAC World Congress*, Milan, Italy, Aug. 2011.
[5] B. Chazelle, "A theorem on polygon cutting with applications," in *23rd Annual Symposium on Foundations of Computer Science (SFCS '08)*, Nov. 1982, pp. 339 –349.
[6] F. Chang, C. jen Chen, and C.-J. Lu, "A linear-time component-labeling algorithm using contour tracing technique," *Comput. Vis. Image Underst*, vol. 93, pp. 206–220, 2004.
[7] E. Hernández, P. Ridao, A. Mallios, and M. Carreras, "Occupancy grid mapping in an underwater structured environment," in *Proceedings of the 8th IFAC International Conference on Manoeuvring and Control of Marine Craft (MCMC)*, Guaruja, Sao Paulo, Brazil, Sep. 2009.