2011 IEEE International Conference on Robotics and Automation
Shanghai International Conference Center
May 9-13, 2011, Shanghai, China

A Topologically Guided Path Planner for an AUV Using Homotopy Classes

Emili Hernández, Marc Carreras, Javier Antich, Pere Ridao and Alberto Ortiz

Abstract— **The paper proposes a method that uses topological information to guide the path search in any 2D workspace. As a first contribution, we have extended the method proposed by Jenkins to generate the topological information by taking into consideration the constraints of the workspace during the construction of the topological environment, which is then used to compute topological paths. As a second contribution, a planner based on the Rapidly-exploring Random Tree (RRT), called *Homotopic RRT* (HRRT) is proposed to use the topological information to guide the path search in the workspace. Finally, simulated and real results with an Autonomous Underwater Vehicle (AUV) are presented showing the feasibility of the proposal. Comparison with well-known path planning algorithms has also been included.**

I. INTRODUCTION

The work presented in this paper is focused in autonomous underwater robotics. The main goal of the research project is to design a motion system that is able to detect the environment, build a internal local map of the explored area, compute a safe path through the map and, finally, generate the high-level commands to follow it. The motion system has to be able to update the map and the path according to the information obtained from the unknown environment in a reduced amount of time (less than 10-20s). The target application of this research project is autonomous navigation in harbors, ocean observatories, fleets of ships or steep seabeds for inspection, security or maintenance tasks.

This paper addresses the design of the path planning algorithm to generate a path in the local map in a very short time. Our approach assumes that the local map is constructed just for navigation purposes. This kind of applications requires fast path planning. Therefore, anytime planners [1] are the most suitable ones: they find a first solution, possibly highly suboptimal, very quickly and then they refine it until time runs out.

Most of the anytime path planning algorithms require an ϵ value and a decrement factor ([2] and [3]). The ϵ value is a multiplication factor used to control the cost of the generated solution. At each iteration, it is decreased by the decrement factor until $\epsilon = 1$, if time does not expire before. Anytime Repairing A* (ARA*) proposed by Likhachev [4] is a reference within the class of deterministic/heuristic based anytime path planners. It inflates the heuristic function using the ϵ value to guide the search towards those states which are close to the solution whose final cost is ensured not to be more than ϵ times the cost of the optimal path. Although at each iteration the solution is intended to be improved by decreasing ϵ, the generation of a new/better path is not ensured (the same path as in the previous iteration of the algorithm can be obtained), which means a waste of computation time in a critical context since the available time to perform the path planning is very limited.

Recently, Ferguson [5] developed a probabilistic anytime algorithm called Anytime-RRT (ARRT). It is a sampling-based planner that works by generating a series of RRTs where each new tree reuses the cost information from the previous tree to control its growth and thus improve the quality of the resultant path. Notice that, unlike the ARA*, the ARRT scarcely reutilizes data between iterations because each new tree is almost built from scratch.

Topological approaches are another way to tackle the path planning problem. Essentially, this kind of solution works with a graph-based abstraction of the workspace, what makes the environment to be represented by a reduced number of potential states over the aforementioned strategies ([6] and [7]). Visibility graphs-based approaches constitute a well known class of algorithms in this regard [8]. Another set of solutions adds a further level of abstraction by means of the homotopy concept, what leads to working with the so-called homotopy class instead of standard topological paths ([9] and [10]).

This paper proposes the use of homotopy classes to guide topologically a path planning algorithm. Using the topological information, the planner does not have to explore the whole space but the space confined in a homotopy class. If the homotopy classes that most-probably contain the lower cost solutions are known, then the algorithm can generate some good solutions very fast and, therefore, act as an anytime algorithm. Moreover, as it will be explained in the paper, the homotopy classes allow us to reach the goal position by avoiding the obstacles in different manners, which has interest when the robot is surveilling a particular area.

The paper presents an extension of the method proposed by Jenkins [9] to generate homotopy classes that can be followed in any 2D workspace. The extension of the Jenkins approach that allows path planning following topological restrictions constitutes the first and main contribution of the paper. In order to maximize the number of homotopy

This research was sponsored by the Spanish government (DPI2008-06545-C03-03 and -02) and the TRIDENT EU FP7-Project under the Grant agreement No: ICT-248497.

E. Hernández, M. Carreras, P. Ridao are with the Department of Computer Engineering, University of Girona, Spain {emilihb,marcc,pere}@eia.udg.edu

J. Antich and A. Ortiz are with the Department of Mathematics and Computer Science, University of the Balearic Islands, Spain {javi.antich,alberto.ortiz}@uib.es

classes that can be explored, the proposed planner, called *Homotopic RRT* (HRRT), is based on the RRT algorithm which has been shown to be very efficient in time, even in complex workspaces ([11] and [12]). The HRRT algorithm does not require the setting of the ϵ nor decrement value. Instead of starting the search with a highly suboptimal path that is improved over time, it starts looking for a path in each homotopy class that has high probability of containing the optimal solution. The method is proved to be complete because in case the goal is not reachable, no homotopy classes will exist and, consequently, no paths will be generated. On the other hand, it is important to note that the homotopy class of the global optimal path is guaranteed to be generated by the algorithm. Details and results in simulation about the HRRT algorithm constitute the second contribution of the paper. Finally, in order to evaluate the feasibility of the work in a motion system of an Autonomous Underwater Vehicle (AUV), the paper presents the results obtained with an underwater robot. An underwater environment was built to allow the testing of the path planning algorithm in a water tank. The robot moved through the environment, perceiving the obstacles and generating an Occupancy Grid Map (OGM). Then, the HRRT was applied generating a path for each homotopy class on the map. Experimental results with real data constitute the third and last contribution of the paper.

The paper is structured as follows. In section II, describes the Jenkins method to obtain topological paths from a metric environment. Section III details our proposal to extend the Jenkins method. Section IV describes the topological-guided path planning algorithm that we propose. Section V reports the results, and section VI exposes the conclusions and future work.

II. The Jenkins Method

Given a workspace with obstacles, Jenkins [9] proposes a method to generate a topological representation of the environment which is used to extract all the different *homotopic* paths from the starting to the ending point. Basically, two paths are homotopic if they go through the obstacles in the same manner.

Formally, given a region R defined by the workspace, let P be the set of all continuous, obstacle-avoiding paths from a starting point to an ending point. Then, two paths $p_i, p_j \in P$ are *homotopic* if they can be mapped to each other through a continuous function without encroaching any obstacle. The homotopy relation is reflexive, symmetric, and transitive, and forms an equivalence relation. Thus, the homotopic functions in P form equivalence classes, and these are called *homotopy classes* [9].

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 construction method proposed by Jenkins [9] in a scenario with n obstacles starts by choosing a random point for each obstacle, which is labeled as b_k, where $k = 1..n$. Then, a central point c is randomly selected. This point cannot be inside an obstacle nor be inside the $n(n-1)/2$ lines determined by the pairwise choices of distinct b_k. Finally, n lines l_k joining c with each b_k are constructed. Each line is partitioned into two directed semifinite rays: the ray emanating from b_k and away from c is labeled β_k and the ray from b_k that contains c is labeled α_k. Fig. 1a shows an example of a reference frame in a scenario with two obstacles.

Using the reference frame, any path p can be defined by the sequence of names of the rays being crossed in order from the starting to the ending point. Nevertheless, there are two special cases: when p crosses no rays then $p = \emptyset$ and when p crosses through c meaning that all the α's are simultaneously crossed. In such a latter case, all α_k are added in subindex order to the sequence.

It is common practice to work with the *canonical representation* of the path instead of the path itself to obtain the same representation for paths that are homotopic but do not follow the same crossing-ray order in the reference frame. With the nomenclature used by Jenkins, canonical sequences are obtained by sorting the α's substrings in non-decreasing order of subindex and then removing all the pairs of the sequence that have the same character. This process is repeated until no changes are made to the sequence (Alg. 1).

Algorithm 1 Canonical representation

$\mathbf{CanonicalRepresentation}(p)$

1: **repeat**
2: $\quad p_{sorted} \leftarrow Sort\alpha' sSubstrings(p)$
3: $\quad p_{canceled} \leftarrow CancelEqualPairs(p_{sorted})$
4: $\quad p \leftarrow p_{canceled}$
5: **until** $(p_{canceled} = p_{sorted})$
6: **return** p

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. The reference frame divides the metric space into *wedges* which are represented as nodes of G. Each node is connected with its neighbors with one or two edges depending on the segments traversed between wedges. These edges are labeled according to the segment crossed 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. 1 depicts a path in the reference frame and its equivalent description in the topological graph.

Once the topological graph is constructed, Jenkins proposes to traverse it using a modified version of the Breadth-First Search algorithm (BFS) [13] to generate all the homotopy classes that are not self-crossing nor duplicated. For example, in Fig. 1 the homotopy classes would be: $\alpha_2\beta_1\beta_2$, $\alpha_2\beta_1\alpha_2$ (depicted), $\alpha_2\alpha_1\beta_2$, $\alpha_1\beta_2\alpha_2$ and $\alpha_1\beta_2\alpha_1\beta_1\beta_2$.

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

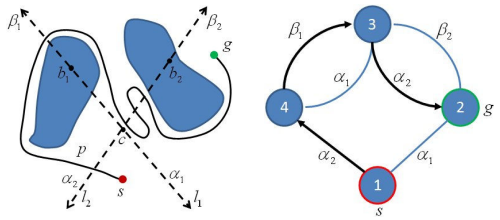


Fig. 1. Topological path: a) in the reference frame ($p = \alpha_2\beta_1\alpha_2\alpha_2\alpha_2$) and b) in the topological graph represented with its canonical sequence ($\alpha_2\beta_1\alpha_2$) .

C. Applicability to the path planning problem

We propose to use the Jenkins method to guide a path planning algorithm following a topological path. Thus, the topological information of the homotopy classes has to be turned into metric paths in the workspace by using the reference frame as a link between the topological graph and the workspace. Therefore, in order to find a path in the workspace that follows a specific homotopy class, it is required a path planning algorithm conveniently modified to look for the intersections with the desired segments of the reference frame.

During the reference frame construction, each obstacle of the map is represented by a b_k point without area in order to ensure that each line of the reference frame only crosses one obstacle; and the topological graph is built under this assumption. However, depending on the reference frame construction and the particular shape of the obstacles it is possible that a line of the reference frame intersects with more than one obstacle. In some cases, there will be homotopy classes that cannot be followed in the workspace. Fig. 2 depicts this problem: assuming that the homotopy class to follow is $\beta_1\alpha_2$, in Fig. 2a it is shown a path in the reference frame with its equivalence in the topological graph Fig. 2b. However, in Fig. 2c the metric path cannot be followed because obstacle 2 is crossing l_2 and l_1. Notice that this problem would not arise if points c and b_1 were more carefully selected to avoid the intersection of the obstacle 2 with l_1. However this solution cannot be applied in complex scenarios with more obstacles.

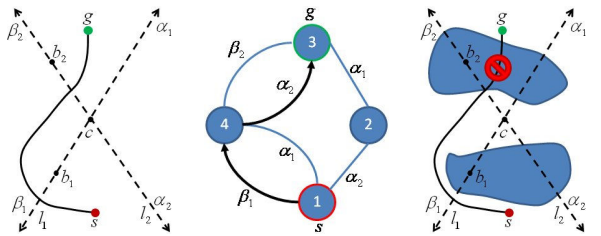


Fig. 2. Example of a valid homotopy class ($\beta_1\alpha_2$) in the reference frame a) and in the topological graph b) that cannot be followed in the workspace c) because at least one line (l_1 or l_2) in the reference frame intersects more than one obstacle.

III. Extension of the Jenkins method

In order to avoid the problem bound in section II-C, this paper proposes an extension of the Jenkins method which takes into account the shape of the workspace obstacles during the construction of the reference frame and the creation of the topological graph. The homotopy classes would be computed according to the intrinsic restrictions of the scenario keeping the coherence between the topological path and its metric representation in the workspace.

A. Reference Frame

As in Jenkins method, each obstacle k has to be represented as a single point b_k. However, it is necessary to distinguish different segments when other obstacles intersect line l_k. Thus, we extend the notation of the segments to describe this fact. The whole construction process can be summarized in three steps:

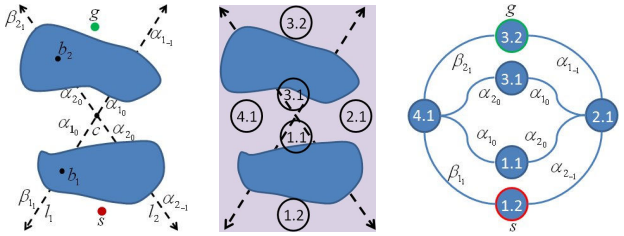
1) Select a random point for each obstacle and label it as b_k, where $k = 1..n$.
2) Select the central point c of the reference frame. This point cannot be inside an obstacle nor being inside the $n(n-1)/2$ lines determined by the pairwise choices of distinct b_k.
3) Construct n lines l_k joining c 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 β_{k_s}, and the segments in the opposite direction are labeled α_{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.

This kind of labeling allows us to distinguish different segments created by the l_k-obstacles intersections in the workspace. Fig. 3a depicts the new reference frame for the scenario in which we had problems with the original Jenkins topological description (section II-C).

B. Topological Graph

The method to construct the topological graph G has been modified to take into account the extended reference frame. Basically, 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.



a) Reference frame b) Regions c) Topological graph

Fig. 3. Example of a reference frame a) the regions created by the intersections of the lines l_k and the obstacles b) and the correspondent topological graph c) using the extension of Jenkins method.

As can be seen in Fig. 3b and Fig. 3c, now the topological graph takes into account the visibility between regions according to the representation of the obstacles in the reference frame which contains the restrictions of the workspace intrinsically. Notice that any topological path of the topological graph can now be followed in the workspace. Hence, the homotopy class $\beta_1\alpha_2$, which is represented $\beta_{1_1}\alpha_{2_0}$ with the new notation, is not valid anymore to describe a path from s (node 1.2) to g (node 3.2).

C. Generation of Homotopy Classes

The topological graph is traversed using the version of the BFS algorithm proposed by Jenkins [9]. Basically, 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, and so on, until it finds the goal. Unlike the standard BFS, which stops when all vertexes have been visited, the algorithm proposed by Jenkins 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.

D. Restriction Criteria

The restriction criteria 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:

1) Wrap: Any string that contains a substring of the form $\alpha_{k_s}...\chi_{k_t}...\alpha_{k_s}$ or $\beta_{k_s}...\chi_{k_t}...\beta_{k_s}$ where $\chi = (\alpha, \beta)$ represents a class that wraps around an obstacle and is self-crossing.

2) Self-crossing: Without taking into account the sub-segment division, any string that in its canonical sequence contains a substring of the form $\alpha_k...\beta_m...\alpha_m...\beta_k$ or $\beta_k...\alpha_m...\beta_m...\alpha_k$ is a candidate to be self-crossing. To ensure there is a self-cross, the sequence of nodes of the topological graph, traversed when the string is generated, has to contain a cycle. If this cycle is made by the starting node, it is not ensured that the path self-crosses. Otherwise, it is guaranteed to be self-crossing and the candidate homotopy class is discarded.

3) Duplicated: Duplicated strings are not allowed in the list of homotopy class candidates. To check whether two strings are duplicated, their α's substrings should be previously sorted as described in Alg.1. Moreover, any string that in an α's substring has duplicated terms is subjected to a cancelation function as reported in Alg.1. After applying it, the resultant string is ensured to be a duplicate of another homotopy class already computed. 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, as stated before, the pair would be canceled and the string discarded for being duplicated.

IV. 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. Our proposal is a probabilistic planner based on the goal-biased RRT algorithm which has been shown to be effective for solving path planning task in many kinds of problems (see [14] and [12]). It is called *Homotopic RRT* (HRRT) and just allows a constrained growing of the tree in those directions that satisfy a given homotopy class. Before adding a new node into the tree, it is checked that the topological path traversed from the starting configuration to the node is contained in the homotopy class by computing the intersections of the path with the reference frame.

The algorithm is detailed in Alg. 2. It receives as input parameters the start and goal configuations (q_{start} and q_{goal}), a distance threshold ($distThreshold$), a candidate homotopy class to follow ($tPath$) and the reference frame ($refFrame$). The nodes of the tree T 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 RRT, the function $Extend$ (line 31) iteratively extends the tree T until the distance between the configuration of n_{new} ($Q(n_{new})$) and n_{goal} ($Q(n_{goal})$) is lower than a given threshold. In this function, $ComputeQRand$ selects a random configuration q_{rand} from the workspace. Then, the $NearestNeighbor$ function returns the nearest node $n_{nearest}$ regarding a random configuration q_{rand} by looking for the node whose topological path is closer to $P(n_{goal})$ (line 4). If there is more than one candidate, the node selected is the closest to the goal according to the Euclidean distance. After q_{new} is computed using the function $ComputeQNew$, $FindIntersections$ (line 21) checks whether the segment $[Q(n_{nearest}), q_{new}]$ intersects with any segment of the reference frame F[2]. The function returns the intersected edges sorted by distance from

[2]Notice that it is possible to intersect with more than one segment of the reference frame depending on the step between $n_{nearest}$ and q_{new} and how close these nodes are to the c point.

$Q(n_{nearest})$. Then the function $UpdatePath$ (line 22) generates the new topological $path$ according to the intersections. No intersection with F means that the tree grows in the $n_{nearest}$ $wedge$ and hence, the function returns $P(n_{nearest})$. If there are intersections and these intersections follow the topological path, the function returns $P(n_{nearest}) \cup I$ in order to create a candidate new node n_{new} to be added to the tree; otherwise a $null$ path is returned and no node is added to the tree.

Algorithm 2 Homotopic RRT

NearestNeighbor(T, q_{rand})
1: $n \leftarrow T$; $d \leftarrow Distance(Q(n), q_{rand})$
2: **for all** $c \leftarrow T.Children()$ **do**
3: $[n', d'] \leftarrow NearestNeigbor(T, q_{rand})$
4: **if** $(|P(n')| > |P(n)|)$ **or** $(|P(n')| = |P(n)|$ **and** $d' < d)$ **then**
5: $n \leftarrow n'$; $d \leftarrow d'$
6: **end if**
7: **end for**
8: **return** $\{n, d\}$

FindIntersections$([q_{nearest}, q_{new}], F)$
9: $r \leftarrow \emptyset$
10: **for** $i \leftarrow 1$ **to** $|F|$ **do**
11: **if** $p \leftarrow Intersection([q_{nearest}, q_{new}], F[i]) \neq null$ **then**
12: $r \leftarrow r \cup \{Edge(i), Distance(q_{nearest}, p)\}$
13: **end if**
14: **end for**
15: $r \leftarrow SortByDistance(r)$
16: **return** r

Extend(T, n_{goal}, F)
17: $n_{new} \leftarrow \{\infty, null\}$
18: $q_{rand} \leftarrow ComputeQRand()$
19: $n_{nearest} \leftarrow NearestNeighbor(T, q_{rand})$
20: $q_{new} \leftarrow ComputeQNew(Q(n_{nearest}), q_{rand})$
21: $I \leftarrow FindIntersections([Q(n_{nearest}), q_{new}], F)$
22: $path \leftarrow UpdatePath(P(n_{nearest}), I\}$
23: **if** $(path \neq null)$ **then**
24: $n_{new} \leftarrow \{q_{new}, path\}$
25: $n_{nearest}.Add(n_{new})$
26: **end if**
27: **return** n_{new}

HRRT()
28: $n_{new} \leftarrow \{q_{start}, \emptyset\}$; $n_{goal} \leftarrow \{q_{goal}, tPath\}$
29: $T.Add(n_{new})$
30: **while** $Distance(Q(n_{new}), Q(n_{goal})) > distThreshold$ **do**
31: $n_{new} \leftarrow Extend(T, n_{goal}, refFrame)$
32: **end while**

V. Results

The extension of the Jenkins method and the path planning algorithm we propose 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 [15]. 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.

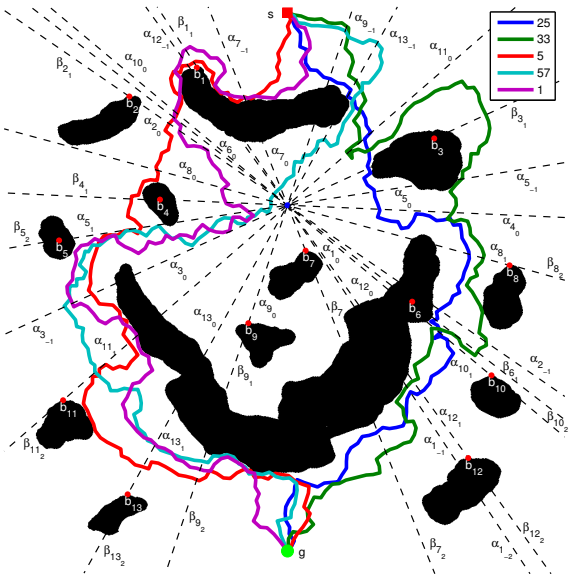


Fig. 4. Paths of the five homotopy classes whose paths are closer to optimal. The class associated to the index can be found in Tab. I.

A. Simulated Results

Fig. 4 depicts the bitmap of 1000x1000 pixels with 13 irregular obstacles used to test the extension of the Jenkins method and the HRRT algorithm. The construction of the reference frame, the topological graph and the generation of 112 homotopy classes took $0.72s$. Fig. 5 depicts the normalized cost and the computation time for each homotopy class sorted by the quality of its solution. To ensure the stability of the results, each path has been generated 50 times and the figure represents the average of them.

Although it is difficult to compare our proposal because, to best of the authors' knowledge, there is no other probabilistic sampling-based path planning algorithm that guides the search topologically, we have implemented the A*, the RRT and their respective anytime versions (ARA* and ARRT) to enhance the comparison. As a result of the HRRT, we have chosen the five homotopy classes whose paths cost are closer to the optimal solution. These classes are listed in Tab. I and their paths are depicted in Fig. 4.

Idx	Homotopy class
25	$\alpha_{9_{-1}}\alpha_{13_{-1}}\alpha_{11_0}\alpha_{3_0}\alpha_{5_0}\alpha_{4_0}\alpha_{8_1}\alpha_{2_{-1}}\beta_{6_1}\alpha_{10_1}\alpha_{12_1}\alpha_{1_{-1}}\beta_{7_2}$
33	$\alpha_{9_{-1}}\alpha_{13_{-1}}\alpha_{11_0}\beta_{3_1}\alpha_{5_{-1}}\alpha_{4_0}\alpha_{8_1}\alpha_{2_{-1}}\beta_{6_1}\alpha_{10_1}\alpha_{12_1}\alpha_{1_{-1}}\beta_{7_2}$
5	$\alpha_{7_{-1}}\beta_{1_1}\alpha_{12_{-1}}\alpha_{10_0}\alpha_{6_0}\alpha_{2_0}\alpha_{8_0}\beta_{4_1}\alpha_{5_1}\alpha_{3_{-1}}\alpha_{11_1}\alpha_{13_1}\beta_{9_2}$
57	$\alpha_{9_{-1}}\alpha_{13_{-1}}\alpha_{13_0}\alpha_{9_0}\alpha_{7_0}\alpha_{1_0}\alpha_{12_0}\alpha_{10_0}\alpha_{6_0}\alpha_{2_0}\alpha_{8_0}\alpha_{4_0}\alpha_{5_0}\cdots$ $\cdots\alpha_{3_{-1}}\alpha_{11_1}\alpha_{13_1}\beta_{9_2}$
1	$\alpha_{7_{-1}}\beta_{1_1}\alpha_{12_{-1}}\alpha_{10_0}\alpha_{6_0}\alpha_{2_0}\alpha_{8_0}\alpha_{4_0}\alpha_{5_0}\alpha_{3_{-1}}\alpha_{11_1}\alpha_{13_1}\beta_{9_2}$

TABLE I
THE FIVE HOMOTOPY CLASSES WHOSE PATHS HAVE THE LOWER COST WITH THEIR INDEX.

The results of the RRT and ARRT planners are the average of 100 executions. As shown in Fig. 6, the RRT algorithm took $0.195s$. with a cost of 1.51 times the optimal path. The ARRT took more than $3s$ to obtain the first solution,

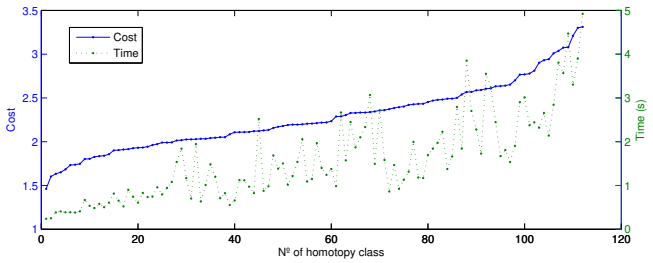


Fig. 5. Normalized cost and computation time for paths generated with the HRRT for each homotopy class. Each point represents the average of 50 iterations.

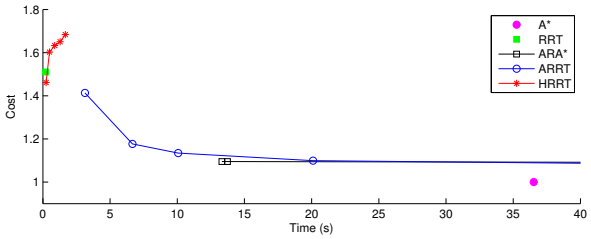
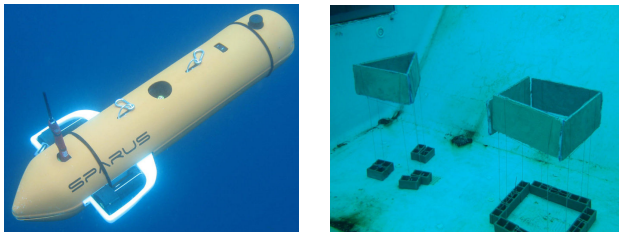


Fig. 6. Comparison of the HRRT paths of the best five homotopy classes vs A*, RRT, ARA* and ARRT algorithms.

and $78s$ to obtain all of them, but it ensured that any new generated solution was closer to the optimal one. The A* returned the optimal path in $36.53s$ and the ARA* generated the first solution in $13.32s$ and found the optimal solution after $775s$. The optimal solution computed by the A* was used to normalize the costs presented in Fig. 5 and Fig. 6. The HRRT computed the best solution (index 25) in $0.24s$ and obtained the path for the five selected homotopy classes in $1.67s$. Notice that this time was required to compute the best five homotopy classes, which are not known in advance. (Future work will consist in finding a quantitative measure for homotopy classes to estimate their quality before computing their path). The HRRT computation time was better than the deterministic approaches and the ARRT. The reduced computation time of the RRT could not be reached due to the extra load of checking the topological restrictions. On the other hand, the best homotopy class (index 25) had a lower cost than the RRT.

B. Experimental Results

The algorithm described in this paper has been tested with the SPARUS^{AUV} (Fig. 7.a). The robot was designed to participate at the Student Autonomous Underwater Vehicle Challenge Europe (SAUC-E) 2010 edition organized by NATO Research Centre (NURC) in La Spezia, Italy. The SPARUS^{AUV} is a $35Kg$ torpedo shaped vehicle of $1.22m$ length x $0.23m$ diameter whose motion is controlled by three Seabotix thrusters. It is equipped with an embedded computer with an Intel® Core™ Duo Processor U2500@$1.2GHz$. Its sensor suite is composed by a forward-looking and down-looking color video cameras, a MTi Motion Reference Unit (MRU) from XSens Technologies, a Micron Mechanical Scan Imaging Sonar (MSIS) from Tritech, an echosounder, a pressure sensor, a Doppler Velocity Log



a) b)

Fig. 7. SPARUS^{AUV} (a) and a detail of the underwater set up (b).

(DVL) from LinkQuest which also includes a compass/tilt sensor and several temperature, voltage, pressure sensors and water leak detectors for safety purposes.

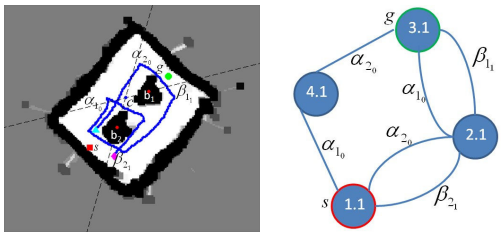
The experiment took place in the Underwater Robotics Lab. of the University of Girona. In order to put obstacles of different shapes and sizes stacked in the water tank, we built a set of plates, each one made of a 1.20x0.60x0.04m roof insulator panel covered with a plastic mesh and concrete in both sides. The insulator is a cheap alternative to wood which offers buoyancy and the concrete adds weight and allows to simulate harbors texture. Notice that none of the materials of the panels are ferromagnetic, hence the compasses of the vehicle are not affected while navigating at close distance.

For the experiment, a triangular and a squared obstacles where set up. Each panel was stack at $3m$ depth using weights (Fig. 7.b). The MSIS was configured to scan the whole 360° sector and it was set to fire up to a $5m$ range with a $0.1m$ resolution and a 1.8° angular step. The trajectory of the robot is based on dead-reckoning, computed using the velocity readings coming from the DVL and the heading data obtained from the MRU sensor, both merged with an EKF using a constant velocity model with acceleration noise. To avoid perturbations on the DVL measurements, the test zone was limited in the maximum depth area, which is at the centre of the pool.

During the experiment, the robot was teleoperated through the obstacles as shown in the trajectory of Fig. 8.a. While navigating at $3m$ depth, the vehicle was building a 18x16m OGM with a $0.1m$ resolution using the navigation from the EKF and the beam information received from the MSIS [16]. The inverse sensor model of the MSIS was implemented as a modified version of a regular sonar sensor with an overture of 3°. Fig. 8.a also depicts the C-Space based on the OGM, the obstacles identified by the CL algorithm, the reference frame and its topological graph (Fig. 8.b). Using the modified version of the Jenkins method, four homotopy classes were generated. Fig. 9 shows the homotopy classes with their path lengths. Each path is depicted with its corresponding tree generated with the HRRT algorithm. The process of applying the CL algorithm, the reference frame and topological graph construction and the generation of the paths in the C-Space using the HRRT took less than $100ms$.

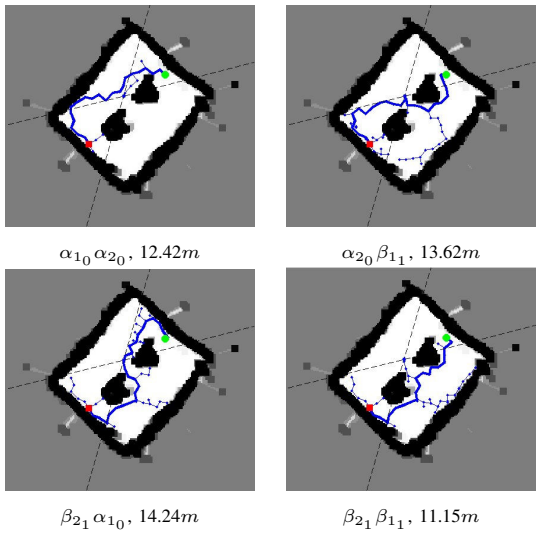
VI. Conclusions and Future Work

This paper proposes a method to guide path planning algorithms with topological information. Given a map with



a) b)

Fig. 8. Trajectory of the robot and the reference frame plotted in the 18x16m C-Space with 0.1m resolution (a) and its topological graph (b).



$\alpha_{1_0}\alpha_{2_0}$, 12.42m $\alpha_{2_0}\beta_{1_1}$, 13.62m

$\beta_{2_1}\alpha_{1_0}$, 14.24m $\beta_{2_1}\beta_{1_1}$, 11.15m

Fig. 9. Paths and trees computed with HRRT using the generated homotopy classes. In the plot, the squared and the circular points correspond with the start and goal points.

obstacles, we first use an extension of the Jenkins method to construct a reference frame which allows to represent any path in the workspace as a topological sequence. The extension of the Jenkins method constitutes the main and first contribution of this paper. As a second contribution, we have developed the HRRT, a path planning algorithm that generates paths in the workspace following the homotopy classes previously found. The effectiveness of the algorithm has been shown in a big and relative complex simulated scenario. As a third contribution of this paper, our proposal has also been tested in real conditions with an underwater robot in a controlled unknown environment to test its applicability to real applications. While navigating, the robot was generating the C-Space based on an OGM, built using navigation and MSIS data. The robot identified the obstacles of the scenario with the CL algorithm, applied the modified Jenkins method to generate the reference frame, the topological graph, and generated the path for each homotopy class using the HRRT algorithm in less than 100ms, which is enough to accomplish our robot realtime specifications. Future work will consist in using the paths generated by the HRRT to guide the robot autonomously. The algorithm will allow generating new paths each time that substantial changes will be detected in the OGM.

The HRRT algorithm we propose, is based on an implementation of the RRT that guides the random sampling towards the goal without considering that it has to follow a particular homotopy class. The current implementation of the HRRT just prevents the tree to grow out of the regions of the reference frame that do not follow the homotopy class. Therefore, we think it is possible to improve the performance and the quality of the solution of the HRRT by guiding the random sampling towards the regions of the reference frame that follow the homotopy class.

Finally, as stated in section V-A, future work will consist in finding a quantitative measure for homotopy classes to get an estimation of their quality before computing their path with the HRRT in order to improve the effectiveness of the whole process.

REFERENCES

[1] D. Ferguson, M. Likhachev, and A. Stentz, "A guide to heuristic-based path planning," in *Proceedings of the International Workshop on Planning under Uncertainty for Autonomous Systems, International Conference on Automated Planning and Scheduling (ICAPS)*, June 2005.

[2] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic A*: An anytime, replanning algorithm," in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, June 2005.

[3] D. Ferguson and A. Stentz, "Anytime, dynamic planning in high-dimensional search spaces," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2007.

[4] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," in *In Advances In Neural Inforamtion Processing Systems 16: Proceedings of the 2003 Conference (NIPS-03)*. MIT Press, 2004.

[5] D. Ferguson and A. Stentz, "Anytime RRTs," in *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '06)*, October 2006, pp. 5369 – 5375.

[6] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Robotic exploration as graph construction," *Robotics and Automation, IEEE Transactions on*, vol. 7, no. 6, pp. 859 –865, dec 1991.

[7] E. Fabrizi and A. Saffiotti, "Extracting topology-based maps from gridmaps," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2000, pp. 2972–2978.

[8] J. Latombe, *Robot Motion Planning*. Boston, MA: Kluwer Academic Publishers, 1991.

[9] K. D. Jenkins, "The shortest path problem in the plane with obstacles: A graph modeling approach to producing finite search lists of homotopy classes," Master's thesis, Naval Postgraduate School, Monterey, California, June 1991.

[10] S. Cabello, Y. Liu, A. Manler, and J. Snoeyink, "Testing homotopy for paths in the plane," in *Proceedings of the Symposium on Computational Geometry (SoCG)*, 5-7 June 2002.

[11] K. J. J. Lavalle S. M., "Randomized kinodynamic planning," in *Proceedings of the IEEE International Conference on In Robotics and Automation (ICRA)*, vol. 1, September 1999, pp. 473–479.

[12] J. Kim and J. Ostrowski, "Motion planning a aerial robot using rapidly-exploring random trees with dynamic constraints," in *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, vol. 2, September 2003, pp. 2200–2205 vol.2.

[13] D. E. Knuth, *The Art Of Computer Programming*, 3rd ed. Addison-Wesley, 1997, vol. 1.

[14] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with RRTs," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2006, pp. 1243 – 1248.

[15] 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.

[16] 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*, Guaruja, Sao Paulo, Brazil, September 2009.