A Topological Map Based Approach to Long Range Operation of An Unmanned Surface Vehicle

Aditya S. Gadre, Shu Du, Daniel J. Stilwell

Abstract—We present an approach to planning dynamically feasible vehicle trajectories for applications where the vehicle operates in very large environments and for which a kinematic model is a poor approximation of the vehicle’s dynamics. Our approach is based on new methods for generating topological maps of a sparse and natural environment, such as a tropical riverine system.

I. INTRODUCTION

We address the challenge of guidance for unmanned surface vehicles (USVs) that operate in very large and poorly mapped natural environments. Our work is motivated by applications in which an autonomous vehicle must operate in tropical riverine systems, which can be poorly mapped due to changes in the river topography, such as flooding, and the existence of a multi-layer canopy that can obscure the waterway in satellite imagery. We presume a vehicle may travel hundreds of kilometers through a natural environment, and thus an efficient representation of the environment is required in order to facilitate real-time planning.

Our principal contribution is a method for planning dynamically feasible vehicle trajectories based on a topological representation of a sparse, natural environment. We describe a method for creating a topological map along with methods for repairing the topological map in real-time based on sensor data. Our solution also addresses the fact that vehicle dynamics are important. USVs, for example, display significant side-slip, especially at low speeds. Thus guidance solutions that presume a kinematic model may generate unacceptable vehicle trajectories. In our approach, we generate dynamically feasible trajectories given dynamics of the vehicle, including side-slip behavior. Our approach is divided between global planning and local planning. Global planning is based on a topological map of the environment while local trajectory generation is based on a metric map of the local environment.

The work presented here extends the work in [1], in which a formal receding horizon approach to generate dynamically feasible trajectories was described and applied to USV operating in a riverine environment. However, environments considered in [1] were of such sizes that they could be represented by an occupancy grid map of the operating area. In this work, riverine environments are assumed to be potentially very large which prohibits the use of metric maps for representing global information and hence we propose the use of topological map to abstract the environment instead.

In our approach, the vehicle begins by defining a small local area around itself in which to plan locally dynamically feasible trajectories. The local area is represented by an occupancy grid map, which is continuously updated using real time data from onboard sensors. A local goal in the local area is obtained from the global topological map, and dynamically feasible trajectories towards the local goal are computed continuously in the receding horizon framework. Topological map is repaired only when there is a meaningful mismatch between locally sensed information and the global topological map. We adapt the matching condition test in [1] to formally determine when there is a mismatch between local and global information, and thus when the global topological map needs to be repaired. This allows for postponing global updates to the topological map until absolutely necessary.

This paper is organized as follows. We briefly discuss related work in Section II, and introduce topological map and roadmap in Section III. Path planning using roadmap is described in Section IV and Section V details procedure for repairing the topological map. Simulation results are presented in Section VI.

II. BACKGROUND

Environmental map building approaches have been broadly divided into two major paradigms, metric approaches and topological approaches. A metric map, also known as grid map or occupancy map, is relatively easy to construct and maintain from sensor measurements. The resolution of a grid map directly relates to how well the environmental information is represented. On the other hand, topological approaches, which represent an environment by graphs, abstract the environment in a compact representation. The compactness of topological maps allows fast planning and low space and time complexity. However topological maps can possess a coarser representation of the environment than grid maps.

Proposed methods for building topological maps can be broadly categorized into Voronoi diagram based approaches [2], [3], [4], [5] and morphological thinning based approaches [6], [7], [8]. Voronoi diagram based approaches generate a set of points equidistant to obstacles in n-dimensional space. In [2], topological map of the environment is extracted from a metric map by decomposing it into distinct regions, which are mapped into vertices of the topological map. In [3], [4] an approach to building global
A topological map is presented in which local Voronoi-like graphs, computed from segmented range data, are merged into the global topological map.

An alternative approach for building topological maps is based on morphological thinning algorithms, which are effective in reducing binary images to thin-line representations which retain significant topological and geometric properties of the image. In the context of topological map building, various proposed approaches [6], [7], [8] rely on extracting a skeleton-shaped pattern from occupancy maps and then extracting salient vertices and edges to build the topological map.

Topological maps are suitable for fast path planning algorithms. Given a topological map, a standard graph search algorithm can be used to plan abstract paths in the topological map. These abstract paths can then be used to plan paths that take vehicle dynamics into consideration [2]. Another graph based approach to path planning involves roadmaps [9], [10]. Roadmaps are graphs in which each vertex is a feasible vehicle pose and each edge is a feasible path connecting two feasible vehicle poses. Standard graph search techniques can be used to find a sequence of edges that connect current pose of the vehicle with the goal pose. This approach however is not suitable for very large dynamic environments due to large computational requirements. We adopt a different approach to generate roadmap from a topological map which allows for real time updates to the roadmap when the topological map is repaired.

III. TOPOLOGICAL MAP AND ROADMAP

In this section we describe how a topological map is built from a-priori information and how a roadmap is obtained from the topological map. We assume that a binary occupancy map or a set of binary occupancy maps covering the entire riverine environment is available. Such a binary map can be obtained, for example, from satellite imagery or topographic maps. We presume that such a binary occupancy map of the riverine environment is inaccurate and may only partially map the area. Figures 1 and 2 show an example of a satellite image of a typical riverine environment, along with its corresponding binary image, which can be used to construct a topological map.

We note that although building topological map for a very large riverine environment is computationally intensive, it is done offline only once before the start of a mission. As described in Section V, repairs to the topological map, however, are accomplished in real time. We also note that the riverine environment in Figure 2 is not a very large riverine environment. However it is used in the following discussions for clarity in illustrations.

Our approach to building topological map is inspired by the method presented in [11] and combines the morphological thinning operation with the distance map computation on the binary input image to extract an undirected graph that constitutes the topological map. A roadmap is then built from the topological map by adding the goal vertex and computing cost-to-go from all vertices in the graph to the goal vertex.

In the following discussion, we denote the binary input image by $I$ and the undirected graph constituting the topological map by $G = (V_g, E_g)$, where $V_g$ and $E_g$ are the vertex-set and the edge-set of the graph, respectively. Each vertex in the graph is associated with a 3-tuple, and the set of all such 3-tuples is $X_g = \{(x_v, y_v, c_v) \mid \forall v \in V_g\}$, where $(x_v, y_v)$ is the planar position of vertex $v$, and $c_v$ is its distance from the nearest obstacle. Each edge in the graph is associated with a cost, $l_e$, which is equal to the Euclidean distance between its vertices, and the set of all edge costs is $L = \{l_e \mid \forall e \in E_g\}$. Let $W_{\text{max}}$ be the maximum width of the navigable region. We assume $W_{\text{max}}$ to be significantly smaller than the size of the riverine environment.

A. Distance Transform Over Binary Input Image

The first step in building topological map from a binary input image involves computing the distance transform over the entire image. The distance transform of a binary image is a morphological operation in which distance to the nearest background pixel is computed for each foreground pixel. In our algorithm, background pixels denote obstacles while foreground pixels denote navigable areas. Thus the distance transform computes distance from the nearest obstacle at every point in the navigable area. The distance transform of image $I$ is denoted $DT(I)$. 
B. Skeletonization of Binary Input Image

Once the distance transform is computed, we build a pixel-wide skeleton of foreground pixels by applying the morphological thinning operation [12]. Since the thinning procedure does not account for actual physical dimensions of the vehicle or the navigable pathways in the environment, we impose C-obstacles [13] by extending the background pixels around obstacles by a distance \( R_{\text{min}} \), which is deemed safe for navigation. The resulting skeleton and the distance transform \( DT(I) \) abstract navigable regions into a compact representation.

The skeleton thus obtained is one pixel wide and connected. We denote the skeleton by \( S \), where \( p_i \) is the \( i^{th} \) node (pixel) in the skeleton. A 3-tuple consisting of the planar position of the node and the minimum distance of the node to the nearest obstacle is assigned to every node \( p_i \in S \). We construct a preliminary graph \( G_p = (V_p, E_p) \) by tracing the skeleton, such that \( V_p = \{ v_i \mid v_i = p_i, \forall p_i \in S \} \) and \( E_p = \{ e_i \} \subseteq \{ (v_k, v_l) \mid v_k, v_l \in V_p, v_k, v_l \in N(v_i) \} \), where \( N(x) \) is the 8-connected neighborhood of \( x \) in the binary input image \( I \).

The preliminary graph \( G_p \) contains many spurious branches due to non-regular shoreline and obstacle boundaries. We prune such branches by using the two criteria outlined in [14]. Very short branches are pruned using a length threshold. A branch is completely removed if its length is shorter than the minimum clearance radius of the branch vertex from where the branch originated. For branches that cannot be pruned using this method, we use the criterion of area difference to prune branch nodes whose effective contribution to the topological information in the graph is insignificant. Figure 3 shows the preliminary graph thus extracted from the binary input image.

![Preliminary graph extracted from the binary input image.](image)

**Fig. 3.** Preliminary graph extracted from the binary input image.

C. Extraction of Topological Map

It has been established [15], [16] that a minimal set of vertices of a binary image shape can be extracted such that the union of disks of minimal clearance radii, centered at those vertices, completely describes the binary image shape. A topological map built this way can be used to recreate occupancy information in the environment with high fidelity. In our approach, instead of extracting a minimal set of vertices by studying coverage of the entire shape, we find the set of vertices by studying coverage of the skeleton alone. We also ensure that for every edge, the straight line connecting the adjacent vertices passes entirely through navigable area. This approach results in a set of vertices that reproduce navigable area around the skeleton with high fidelity, but may not reproduce all minor variations along the shoreline. This drawback is alleviated by the real-time updates from sensor data.

We extract topological map \( G = (V_g, E_g) \) from the preliminary graph \( G_p \) by identifying salient vertices and edges in \( G_p \), and discarding those vertices and edges that add redundant topological and geometric information. For this purpose, we categorize vertices into three sets, based upon the degree of each vertex in the preliminary graph \( G_p \).

\[
E_p = \{ v_i \mid \text{degree}(v_i) = 1 \} \\
O_p = \{ v_i \mid \text{degree}(v_i) = 2 \} \\
B_p = \{ v_i \mid \text{degree}(v_i) \geq 3 \}
\]

Vertices in \( E_p \), \( O_p \), and \( B_p \) are end, ordinary and branch vertices respectively. We define a segment \( S_p \) as the collection of vertices between two branch vertices or two end vertices or a branch vertex and an end vertex. For each segment \( S_p \) of the preliminary graph, we travel from one end of the segment towards the other, determining salient vertices that will become part of the topological map. As new vertices are included in the topological map, corresponding edges are added to it. In order to preserve geometric information of the environment, we fix the maximum distance, \( d_{\text{max}} \), between two neighboring vertices in the topological map. Algorithm 1 lists the pseudocode for extracting branch of the topological map from a branch of the preliminary graph. This procedure is repeated for all segments in the preliminary graph to build the entire topological map.

Figure 4 shows the topological map extracted from the binary input image. A total of 338 vertices and 338 edges constitute the entire topological map. The size of the riverine environment abstracted by the topological map is approximately \( 1500m \times 2700m \), with each pixel representing a \( 1m \times 1m \) area. The total length of travel along the topological map is about \( 5km \). To build an occupancy grid map of this riverine environment, we would need \( 1500 \times 2700 = 4,050,000 \) cells, as compared to 338 vertices and edges of the topological map.

Figure 5 shows the topological map superimposed with the disks of minimum clearance radii. The union of these disks approximates the navigable areas in the environment. Areas beyond the union of disks approximate land or any other obstacles to navigation.

D. Building Roadmap

Let \((x_{\text{goal}}, y_{\text{goal}})\) be the planar location of the goal and \(c_{\text{goal}}\) its distance from the nearest obstacle. We build a roadmap \( G_r = (V_r, E_r) \) by adding the goal vertex to the
topological map, that is $V_r = V_g \cup V_{goal}$. We add the edge $e_{goal} = \{v_{goal}, \text{nearest} (v_{goal})\}$ to the roadmap, where
\begin{equation}
\text{nearest} (v) = \min_{v_i \in \text{overlap}(v)} d(v, v_i)
\end{equation}
\begin{equation}
\text{overlap} (v) = \{v_i \mid d(v, v_i) \leq c_v, d(v, v_i) \leq c_v\}
\end{equation}

Recall that every vertex in the roadmap represents an obstacle-free area and every edge represents an obstacle-free path that connects two obstacle-free regions.

Once the goal vertex is added to the roadmap, we apply Dijkstra’s algorithm to solve the single-source shortest path problem to compute cost-to-go $d_v$ from each vertex in the roadmap. This cost is the Euclidean distance from each vertex, along the shortest path, to the goal vertex. It is important to note that the straight line paths in the roadmap do not account for vehicle dynamics. We also note that roadmap must be recomputed when topological map is repaired or goal location changed.

In the next section we describe how topological map and roadmap are used to plan dynamically feasible vehicle trajectories.

\section{Path Planning}

In our approach, a sequence of dynamically feasible trajectories is continuously computed in a local area around the vehicle. Local area is represented by a grid map which is continuously updated using the on-board sensor data.

\subsection{Local Map}

A rectangular local area of dimension $X \times Y$ is defined keeping the vehicle at its center. Let $R_{min}$ be the minimum distance allowed between the vehicle position and the nearest edge of the local area. A new local area is defined when the vehicle gets within $R_{min}$ from the nearest edge. We ensure that $R_{min} \geq R_{sense}$, where $R_{sense}$ is the longest sensor range.

When a new local area is defined, we build an a-priori occupancy grid map $\mathcal{L}$. We take advantage of the minimum clearance radii of vertices in the topological map to approximate the local grid map. We define, for every vertex $v$ in the topological map,
\begin{equation}
D_v = \{p \mid p \in \mathcal{L}, d(p, v) \leq c_v\}
\end{equation}

Let $\text{Conv}(D_{v_i}, D_{v_j})$ be the convex hull of the two disks centered at vertices $v_i$ and $v_j$. Let the set of all vertices within the local area be $V_l = \{v \mid v \in G, (x_v, y_v) \in \mathcal{L}\}$. The approximated empty or navigable area in the local grid map is
\begin{equation}
\text{EMP} (\mathcal{L}) = \bigcup_{v \in V_l} \{\text{Conv}(D_v, D_{v_i}) \mid v_i \in \text{adjacent} (v)\}
\end{equation}

where
\begin{equation}
\text{adjacent} (v) = \{v_i \mid \{v, v_i\} \in E_g\}
\end{equation}
The occupied area in the local map is
\[ \text{OCC}(\mathcal{L}) = \mathcal{L} \setminus \text{EMP}(\mathcal{L}) \] (6)

Figure 6 shows the comparison between the a-priori occupancy grid map approximated from the topological map and the grid map obtained from the satellite image. Black regions indicate land and other obstacles. Navigable empty area in the approximated a-priori grid map is represented by the gray region and is superimposed over the white empty area obtained from the satellite image. As can be seen, empty regions around the image skeleton are reconstructed with high fidelity whereas details about small variations near shoreline farther away from the image skeleton are lost. However, data from onboard sensors removes this discrepancy in real time by continuously updating the local grid map.

\textbf{B. Local Goal}

Once the local map \( \mathcal{L} \) is defined, a local goal is computed using the roadmap. Navigable areas that are connected via navigable waterways may be disconnected in the local map. The vehicle cannot travel from one such region to another without leaving the local area. Let \( CR_p \) be the connected region containing the vehicle. We define the set of vertices within the local area which can be visited by the vehicle without leaving the local area by \( V_c = \{ v \mid v \in V_g, \ v \in CR_p \} \). Let
\[
\begin{align*}
    v_{\text{min}} &= \min_{v \in V_c} d_v \\
    v_{\text{pre}} &= \min_{v \in \text{adjacent}(v_{\text{min}})} d_v
\end{align*}
\]
(7)

The edge \( \{v_{\text{min}}, v_{\text{pre}}\} \in E_g \) thus has one vertex in the local area, within the same connected region as the vehicle, and the other vertex outside the local area. Since all edges in the roadmap form straight line segments, we find the local goal by computing the intersection of this edge with the local area boundary it crosses.

\textbf{C. Local Trajectory Generation}

For the sake of brevity, we briefly describe the process of computing dynamically feasible local trajectories. Details are presented in [1], and the dynamic model of a USV is presented in [17]. Given the vehicle dynamics expressed as
\[ \dot{x}(t) = f(x(t), u(t)), \ x(t_i) = x_o, \ t \geq t_i \] (9)
a sequence of local trajectories are computed that satisfy
\[ u^*_k(t) = \arg\min_{u \in U} \int_{k\delta}^{k\delta+T} q(\tau, x(\tau), u(\tau))d\tau + W_i(x(k\delta+T)) \] (10)
The time \( T \) is chosen so that the trajectory remains within the local area \( \mathcal{L} \). Using a standard receding horizon framework, the control signal \( u^*_k(t) \) is implemented during the time interval \( k\delta \leq t < (k+1)\delta \), where \( \delta < T \), and the control signal \( u^*_{k+1}(t) \) is re-computed at time \( (k+1)\delta \). The portion of \( u^*_k(t) \) corresponding to \( t \in [(k+1)\delta, k\delta+T) \) is discarded.

The terminal cost \( W_i(x) \) in (10) approximates the cost-to-go, estimated from the roadmap, for the vehicle at position \( x \). In [1], a formal test called the matching condition
\[ W_i(x(t_i + T)) - W_i(x(t_i + T - \delta)) \leq -\int_{t_i+T-\delta}^{t_i+T} q(\tau, x(\tau), u(\tau))d\tau \] (11)
is proposed. When (11) is not satisfied, then there is a mismatch between the global roadmap and the local grid map against which the local trajectory is computed. When this happens, we repair the topological map. We do not need to update the topological map otherwise.

\textbf{V. TOPOLOGICAL MAP REPAIR}

Our proposed method divides the topological map update into three steps: removing outdated information from the global topological map, building the local topological map, and finally merging the local topological map with the global topological map. We describe these steps with an illustration in Figure 7.
A. Remove Vertices and Edges inside the Local Area

Due to mismatch between the current topological map and the new local area environment, we first remove the obsolete portion of the current topological map within the local area \( \overline{L} \). The portion of the topological map to be removed forms a subgraph \( G_l = \{ V_l, E_l \} \subset G \). In Figure 7, this subgraph constitutes \( V_l = \{ v_{i-3}, v_{i+3} \} \subset V_o \) and \( E_l = \{ e_{i+2}, e_{i+3} \} \subset E_o \). We define the repaired topological map, after local vertices are removed, by \( G_r = G \setminus G_l \).

Henceforth for brevity of expression, we will use \( v \in \overline{L} \) to mean a vertex \( v \) in some graph \( G = \{ V, E \} \), which satisfies \((x_v, y_v) \in \overline{L} \). Association of vertex with graph will be contextual.

B. Build the Local Topological Map

To build the local topological map, we make use of the property that morphological thinning operation on similar shapes creates similar skeletons. Instead of computing the local topological map only within the local area \( \overline{L} \), we build the local topological map within an extended area \( \overline{L}_e \) containing \( \overline{L} \), and then extract part of this topological map inside \( \overline{L} \).

In order to extract the topological map in \( \overline{L}_e \), we build an extended grid map over \( L_e \) by combining the grid map over \( \overline{L} \) and \( \overline{L}_e \setminus \overline{L} \). However, since no sensor data is available in \( \overline{L}_e \setminus \overline{L} \), we approximate the grid map in this region by using the procedure described in Section IV-A. The local topological map is then extracted using the procedure described in Section III-C. In Figure 7, we indicate the topological map in the extended area by \( G_e = \{ V_e, E_e \} \), where \( V_e = \{ v'_1, v'_2, \ldots, v'_8 \} \) and \( E_e = \{ e'_1, e'_2, \ldots, e'_7 \} \). Since no new sensor data is available outside \( \overline{L} \), parts of the local topological map and the current topological map in \( \overline{L}_e \setminus \overline{L} \) are topologically and geometrically similar.

C. Connect the Local Topological Map to the Global Topological Map

We extract subgraph \( G'_l = \{ V'_l, E'_l \} \) from \( G'_e \) that lies within \( \overline{L} \) and add it to the topological map \( G_r \). For Figure 7, we can write \( G'_l = \{ \{ v'_4, v'_5 \}, \{ e'_5 \} \} \). In order to attach \( G'_l \) to \( G_r \), we make use of the property that image skeletons, and by extension topological maps, are similar for similar binary shapes. Let \( E'_o \subset E'_e \) be the set of edges whose one vertex lies in \( \overline{L} \) and the other in \( \overline{L}_e \setminus \overline{L} \). Since the topological map is an undirected graph, we assume, without loss of generality, that the vertex in \( \overline{L} \) is the source vertex of the corresponding edge while the vertex in \( \overline{L}_e \setminus \overline{L} \) is its target vertex. Algorithm 2 lists the pseudocode for connecting the local topological map \( G'_l \) to \( G_r \).

Following the procedure in Algorithm 2 on the illustration in Figure 7, we connect vertex \( v'_3 \) to \( v_{i+2} \) and vertex \( v'_5 \) to \( v_{i+5} \) to complete the topological map update. Figure 8 shows the result of the topological map update operation.

Once the topological map is updated, the old roadmap is discarded and a completely new roadmap is built from the repaired topological map.

**Algorithm 2: Connect local topological map**

**Require:** \( G'_e = \{ V'_e, E'_e \} \)

**Require:** \( G_r = \{ V_r, E_r \} \)

**Require:** \( E'_o \)

**for all** \( e \in E'_o \) **do**

\( v_{e_i} \leftarrow \text{source}(e) \)

\( v_{e_o} \leftarrow \text{target}(e) \)

\( v_i \leftarrow \text{nearest}(v_{e_i}), v_e \in G'_e, v_i \in G_r \)

\( v_o \leftarrow \text{nearest}(v_{e_o}), v_o \in G'_e, v_o \in G_r \)

\( E_r \leftarrow E_r \cup \{ v_i, v_o \} \)

**end for**

VI. SIMULATION RESULTS

In this section, we present simulation results that demonstrate the topological map repair. To illustrate how repair happens in real-time in response to locally sensed-data, we assume that the binary image used to extract the a-priori topological map is partially wrong. Specifically, as shown in Figure 9, a part of the shoreline contained in the red rectangle is removed.

As shown in Figure 10, we simulate the on-board USV sensor data, resulting in a new shoreline within the current local region. The green rectangle represents the current

![Fig. 8. Final result of topological map update.](image)

![Fig. 9. Wrong binary input image of the riverine environment.](image)
local region and the remainder of the figure represents the extended region in which we compute the local topological map. The red curve which passes over the newly discovered shoreline represents part of the a-priori topological map, and the blue curve represents the local topological map that has been repaired. The repaired local topological map and the a-priori topological map differ significantly only around the newly found shoreline, but are similar everywhere else. Figure 11 shows the repaired local topological map along with the disks of minimum clearance radii superimposed on the extended region.

The repaired topological map thus encodes the current occupancy map information. The roadmap must also be repaired following the topological map repair. We note that the topological map needs to be updated only within a small local region, and thus can be repaired in real time.

VII. ACKNOWLEDGMENTS

The authors gratefully acknowledge support of the Office of Naval Research via grant N00014-10-1-0185.

REFERENCES