Frontier-based Exploration for Unknown Environments Using Incremental Triangulation

Amna AlDahak, Member, IEEE, Lakmal Seneviratne, Member, IEEE, and Jorge Dias, Member, IEEE,

Abstract—In this article we present a new exploration strategy to reduce searching time while simultaneously increasing the robot’s knowledge about its surroundings. A robot’s ability to successfully complete a required task is bound by its knowledge about the operation environment. Thus, the robot must be able to collect information from its surrounding and map it accurately to create a correct representation of the environment and use it as a return path later. An increasing number of applications, such as surveillance and search and rescue, impose time constraints on the mapping and exploration process. The proposed Triangulation-Based exploration maps the environment incrementally using our Incremental Triangulation Algorithm to accelerate the searching process. To achieve those objectives this article introduces the Dynamic Triangulation Tree (DTT) which is a tree-like structure built incrementally during the Incremental Triangulation Algorithm. The DTT compactly represents the geometry of the environment which is utilized for minimizing the overall exploration time. Our solution drives the exploration process towards areas that will provide the robot with greater exposure to its surroundings by considering not only the distances to the frontiers but also the currently known surroundings of the environment. The effectiveness of the DTT and the efficiency of the proposed frontier selection process is validated in simulations using various scenarios to demonstrate the main advantages of the proposed DTT, namely ease of construction, compactness and completeness for “search and rescue” and “sample and return” missions.

Keywords—Mapping, frontier-based exploration, triangulation.

I. INTRODUCTION

AUTONOMOUS robot exploration and map building for unknown environments is increasingly becoming a core requirement in a wide range of applications ranging from search and rescue, surveillance, military and other high risk scenarios. Most of these applications implicitly require the mission to be completed in minimum time and with minimal damage risk for the equipments and people. This additional requirement is generally achieved by using an effective evaluation function for the frontier selection step during the exploration process.

As the robot starts exploring its surroundings, it accumulatively builds a partial map of the environment consisting of the currently known areas. The partial map contains borders separating the explored areas of the environment from unexplored areas known as frontiers [1]. Any exploration strategy is evaluated by the completeness of the constructed environment map and the effectiveness of the construction process. The effectiveness is inversely proportional to the total exploration time and the total distance travelled by the robot during exploration. The exploration time can be significantly reduced by carefully selecting the next position for the robot in each exploration iteration. Thus, exploration is mainly driven by successive selection of exploration frontiers, known as frontier-based exploration [1], [2].

In this article we introduce a new exploration strategy. Where the mapping is developed to compactly capture the environment geometry as well as its connectivity, the exploration is specifically oriented toward minimizing the overall exploration time while maximizing the map coverage of the environment. To accomplish this local optimal solutions given the current map of the environment at any time instance. The proposed Dynamic Triangulation Tree, DTT, was initially developed in conjunction with the Gap Navigation Tree, GNT [3]. The GNT is a data structure used for topological mapping [4]. However, the DTT is found to independently progress towards complete exploration of the given environment as shown in this article.

The remaining of this article is structured as follows: Section II presents the background and related research efforts. Section III discusses the proposed solution with detailed algorithms for all of its modules. The results presented in Section IV validate the effectiveness of the proposed mapping and exploration solutions. Finally, Section V draws the conclusion and presents future research directions.

II. RELATED EFFORTS

Frontier-based Exploration strategies found in the literature vary mainly in two aspects, map representation and frontier selection methods. The majority of the exploration approaches maps the environment using an occupancy grid, such as [5]. An overview of different map representations, including occupancy grids, can be found in [3]. Thus, we will present here research efforts related to frontier selection strategies.

The frontier selection method is the second major factor that contributes to the overall efficiency of any exploration strategy. Exploration solutions generally consider two quantities when selecting a frontier, namely cost and gain. Cost is defined as the effort the robot puts to reach a frontier which is typically the distance to that frontier. Gain is the expected coverage achieved by reaching this frontier.

One of the early exploration solutions was presented by Yamauchi in [1]. He presented a frontier-based exploration strategy where the robot was always assigned to the closest frontier. However, this strategy considered only the cost and ignored the gain factor. The mapping and exploration solution...
introduced in [6] uses grid decomposition for the map representation and always selects the closest frontier as the next-best-view. The frontier selection was further tuned by two additional re-checking and map segmentation to avoid revisiting already explored areas. Later solutions used a combination of cost and gain, such as the recent solution in [7]. For a more comprehensive overview of frontier-based exploration strategies see [8] and [2].

In this article, the frontier selection method uses a combination of cost and gain, where the gain is computed based on properties of each frontier at any moment of time during exploration. Those properties, are directly extracted from the information incrementally added in the mapping structure, the DTT. Therefore, the frontier selection approach adopted in this work cannot be directly compared to other existing techniques in the literature since it utilizes the information coded in the mapping structure, the DTT, developed in this study.

III. THE PROPOSED METHOD

This work focuses on exploration and map building in minimal time. Therefore, in this section we start by defining the robot and the operation environment model. The environment will be incrementally mapped geometrically using the DTT which we will introduce briefly here (for more details please see [3]). The data stored in this structure will be utilized for efficient frontier selection to minimize the exploration time and reduce the distance travelled during exploration. In this article we assume that the robot initiates with a self-localization of itself in partial local map and all subsequent robot’s positions are computed relatively to the previous one.

A. Robot and Environment Model

The robot in this work is modeled as an omnidirectional wheeled robot equipped with two laser range scanners, each with a 180 degrees range. This will provide the robot with a 360 degrees sensing range which will be used to produce the visibility region. The robot operates in a two dimensional, simply-connected, continuous environment. In each iteration during the exploration process, the robot acquires the visibility region from its current position \( x \) denoted as \( \text{VisRegion}(x) \). The discontinuities in the reading will be labeled in the mapped visibility region as non-primitive edges, denoted as \( N\text{Pedges} \). The \( N\text{Pedges} \) are the actual frontiers between explored and unexplored spaces.

B. Exploration and Map Building

The Earlier defined \( \text{VisRegion} \) contains both the list of non-primitive edges and the distances mapped from the incoming sensory data. The \( \text{VisRegion} \) will be divided into a series of connected triangles arranged in a simple yet efficient tree structure. Each triangle will encapsulate details about its geometry and its neighboring triangles. There are mainly two types of triangles, primitive and non-primitive. Primitive triangles are those that have in their neighborhood (i.e. adjacent to their edges) either other triangles or known boundary edges. On the other hand, non-primitive triangles are those that are adjacent to unexplored areas around the robot. In other words, a non-primitive triangle contains at least one non-boundary vertex among its three vertices. A non-boundary vertex is one of the two ends of a non-primitive edge formed by discontinuities in the sensory data readings. Each triangle in the tree is connected to its neighboring triangles. Therefore, traversing the tree from a source triangle to a destination triangle will generate a feasible navigation path to follow by either humans or robots. All related notations are illustrated in Figure 1. The non-primitive triangle edges are the actual frontiers. During exploration, when a frontier, \( e \), is selected the robot moves toward it with a constant speed. This action is denoted as \( \text{explore}(e) \) command.

Before we go any further and start presenting our algorithms in details, the proposed mapping and exploration strategy depicted in Figure 2 starts by initializing the map with the first acquired visibility region. After the initialization step the algorithms iteratively selects and explores a frontier using the Select_Frontier algorithm introduced later. Next it acquires the new visibility region, updates the constructed map, and locates any “interesting findings” on the partially constructed map such as targets or features. A target is any object the robot might be searching for during the exploration. The feature could be an interesting observation or landmark that the system needs to keep a record of by locating it on the map, such as a fire source in fire search and rescue scenarios. In both cases,
the location of the finding is defined on the current partial map and is assigned to the triangle(s) containing it. Using the triangles data structure, the DTT, the system can generate a path to the location of the target/feature in order to reach them despite the incompleteness of the exploration task. However, this step is not addressed in this paper but is considered for future work.

Next we will briefly introduce the map building and incremental triangulation algorithms. This will be followed by presenting the frontier selection strategy developed to reduce the exploration time while increasing the coverage of the explored areas.

1) Build Map Algorithm: The first algorithm introduced here is the Build_Map algorithm which takes the initial sensory readings as its input (distances information) along with the list of discontinuities in the sensory readings. It starts by initializing the DTT, Dynamic Triangulation Tree, created to capture the geometry of the environment in a tree-like triangulated map structure. The novelty in this structure is the incremental nature of its construction process that utilizes the NPedges in the VisRegion to indicate open ends which represent candidate positions for further exploration. Additionally, it maps the explored surroundings in a compact tree structure that minimizes storage and search requirements.

In each iteration the algorithm constructs the VisRegion from the sensory readings and send it to the Incremental Triangulation procedure to update the DTT structure. Upon the completion of the DTT update, the algorithm calls the Select_Frontier algorithm (detailed later) to choose the next frontier to explore. The selected frontier will be explored until reached. The loop continues until the DTT is complete. The DTT is complete when all its triangles are primitive which implies that there are no non-primitive edges, and thus no unexplored frontiers.

2) Incremental Triangulation Algorithm: The Incremental Triangulation algorithm starts by merging the existing VisRegion with the currently visible non-primitive triangles in DTT. Those are basically the triangles adjacent to unexplored areas. In other words, they are the triangles that have at least one non-primitive edge. After the merging process is complete the vertices of VisRegion(x) are stored in decreasing order of y-coordinate. Two vertices will be connected if they are connectable, i.e. they can see each other and no existing DTT edges intersect the new candidate edge. There are 3 main types of edges. Primitive edge with both the vertices it connects are boundary vertices. Non-primitive edge where at least one of its vertices is a non-boundary vertex and the edge is adjacent to the unexplored areas. Temporary edge (denoted temp) both its vertices are non-boundary vertices, or one only is a non-boundary vertex but the edge is in the interior of the explored area so far, i.e. it is not one of the frontiers. This type of edges is subject for removal in future iterations. Figure 3 shows the initial (left) and final (right) snapshots of the exploration process of a simple unknown environment illustrating the mapped DTT at start and end. The initial snapshot shows the initial triangulation constructed using the first acquired VisRegion from the environment. The initial DTT contained a total of 7 triangles, 5 of which were primitive and 2 non-primitive due to their adjacency to unexplored frontiers, i.e. each contains at least one NPedge.

In the final snapshot, all triangles in the DTT are primitive at this stage which indicates the completion of the exploration process.

C. Frontier Selection

The “appropriate” selection of the next frontier to visit during the exploration process can enhance the overall completion of the task. The meaning of the term “appropriate” may vary depending on the task type. In search and rescue, surveillance and some military operations the main purpose of the mission is to reduce the overall exploration time. The strategy developed in this paper is devised toward reducing the exploration time while ensuring the best coverage possible for the operation environment.

While keeping in mind the limited time constraint, our frontier selection process is formulated using two main parameters, cost and gain. The cost of a given frontier is simply the required travel distance to reach it while gain is the expected information gain if this frontier was selected as the next candidate position. Gain can be predicted using two features. The frontier width which the wider the frontier the more the new information expected upon visiting it. The second feature is the number and width of other frontiers in its proximity. A frontier with some neighboring frontiers in a predefined diameter will have high gain. By considering the neighboring frontiers' width we encourage a sequential exploration. Given that if frontier was found to be the best at the current time step, then there is a high probability that neighboring frontiers will be selected next. The two parameters cost and gain, are defined below:

$$cost(e_i) = dist(x, e_i)$$  \hspace{1cm} (1)

Where dist is a function that returns the Euclidean distance between two locations, \(x\) is the current robot location, and \(e_i\) is the frontier (non-primitive edge) under inspection.

$$gain(e_i) = width(e_i) + \sum_{j=0..n}^{i\neq j} \begin{cases} 0 & dist(e_i, e_j) > \epsilon \\ width(e_j) & Otherwise \end{cases}$$  \hspace{1cm} (2)

The gain of an NPedge \(e_i\) is the width of itself added to it an accumulated value of the widths of its neighboring NPedges that lie within a distance smaller than a predefined value \(\epsilon\) from \(e_i\). The value \(\epsilon\) is adapted according to the environment type. For instance, domestic and office-like
environments will utilize a smaller $\epsilon$ than the environments with large open spaces and corridors. Incorrect choice of $\epsilon$ may result in an increase in the overall exploration time. The quantity $value(e_i)$ is the ratio between the two parameters cost and gain.

$$value(e_i) = gain(e_i) / cost(e_i)$$

(3)

As shown in the last formula, if an $NPedge$ $e$ has a very high gain but is relatively far from the current robot position, i.e. has high cost, its value will be low accordingly. In another scenario, if two $NPedges$ have the same cost their gain will affect the final value of each, and accordingly decides which $e$ is selected for the next explore action. Thus, we can see that in each frontier selection decision, the selected frontier is the locally optimal choice given the currently known map of the environment.

The Select_Frontier algorithm details the frontier selection process. This algorithm receives the $DTT$ structure and computes the cost, gain, and value for each $NPedge$ and eventually returns the best candidate robot position for the next iteration in the exploration process.

Algorithm Select_Frontier(DTT)
Input: The existing DTT.
Output: The selected frontier for exploration, frontier.
1. Initialize $value_{\text{max}}$.
2. for each $e_i \in DTT$
3. do $cost(e_i) = \text{dist}(e_i, \text{robotLocation})$
4. $width(e_i) = \text{width of } e \text{ in the non-primitive triangle } t_i$
5. for each $e_i \in DTT$
6. do $gain(e_i) = width(e_i)$
7. for each $e_j \in DTT \neq e_i$
8. do if $\text{dist}(e_i, e_j) \leq \epsilon$
9. then $gain(e_i) = gain(e_i) + width(e_j)$
10. $value(e_i) = gain(e_i) / cost(e_i)$
11. if $value(e_i) > value_{\text{max}}$
12. then $value_{\text{max}} = value(e_i)$
13. frontier = $e_i$
14. return frontier

IV. EXPERIMENTS

The algorithms have been tested in various environmental setups using a testing simulation environment which was developed in Java to validate the accuracy of the proposed solution. Various test scenarios are included in this section. All the presented simulated environments here are assumed to be 16 meter x 16 meter ($\approx 256m^2$) inclusive of the environments’ boundaries. We start by including simulation scenarios to demonstrate the properties of the Dynamic Triangulation Tree, the $DTT$, namely its compactness, ease of construction, and completeness.

To prove the compactness of the $DTT$ structure we will compare the storage space it requires to the space required by the corresponding occupancy grid for various examples. We choose to compare to this representation since it is one of the very commonly used representations in many exploration and mapping solutions as discussed in Section II. Figure 4 shows the final triangulated map of the same environment presented in Figure 3 containing a total of only 10 triangles while the corresponding grid representation (with an approximately 0.4m x 0.4m cell size) of the same environment containing 670 free cells. This shows the compactness of the $DTT$ in comparison to occupancy grid representation.

Figure 5 provides a more exhaustive comparison between $DTT$ and occupancy grid storage size in varying complexity scenarios. Figure 6 statistically compares the storage size required for the proposed $DTT$ with that of three different grids resolutions small (0.4m x 0.4m), medium (0.5m x 0.5m), and large (0.6m x 0.6m). However, the largest resolution makes it infeasible to reach some areas in the environment where the free space dimensions in smaller than the defined cell size. Moreover, with all grids, despite their varying resolution, completeness is always an issue.

The presented comparison proves that our proposed $DTT$ provides a complete mapping of the environment in a very compact representation. The $DTT$ provides not only geometrical mapping, but also facilitate navigation and can easily generate feasible paths in the environment given its connectivity. Searching the $DTT$ for paths is inherently fast due to its tree-like structure. Moreover, path generation is not bound by the completion of the environment mapping but can be executed at any instance of time using the existing partial map.

Another important characteristics of the proposed mapping besides its compactness, is its completeness. As have been discussed earlier, the $DTT$ continues to expand as long as there are non-primitive triangles in the tree. Thus, the exploration is complete when the number of non-primitive triangles reaches zero indicating that the environment is completely mapped in the $DTT$. Figure 7 depicts the variations in the number of primitive and non-primitive triangles during the mapping and exploration of the last office-like environment shown in Figure 5. After the triangulation of the initial $VisRegion$, the number of the primitive triangles start at approximately 20 and continue to increase gradually. On the other hand, the number of non-primitive triangles start at approximately 10 and fluctuates around that value or below as the mapping progresses and it converges to zero toward the end. Thus, when the number of non-primitive triangles is approaching zero it indicates that the mapping and exploration is near completion.

Next we include an example to illustrate the effectiveness of the frontier selection method presented in this paper. Figure 8 shows six snapshots at times $t_1$ (initial position) to $t_6$ (exploration completed) for a domestic environment simulating a residential apartment floor plan. At $t_1$ the $DTT$ has two $NPedges$ $e_1$ and $e_2$. Since, both detected $NPedges$ have
similar values (same cost and gain) one of them will be picked at random, which was $e_2$ in this experiment. At $t_2$, after $e_2$ was explored, two new NP edges were detected, $e_3$ and $e_4$. Following our frontier selection method, $e_3$ was selected next. At this time, $e_1$ was not selected since it is isolated while $e_3$ has $e_4$ in its proximity resulting in a higher gain. Moreover, given the greater width of $e_3$ when compared to $e_4$, $e_4$ exploration was delayed until $e_3$ was explored. Figure 8 (d) shows the completed explore command of $e_4$ at $t_4$ which is followed by $t_5$ showing the robot backtracking to continue the exploration by exploring the only NP edge at this time, $e_1$. The final snapshot (f) at $t_6$ was taken during the final explore. The triangulation in progress and the traversed path by the robot is displayed in all snapshots during this exploration.

The main purpose of our frontier selection method is to increase the covered area of the environment at any time $t$. This is crucial in a variety of applications such as in search and rescue where the survival chances of victims increases by decreasing the time required to locate them.

As the exploration progresses, the robot need to take successive explore actions after selecting the next frontier in each exploration iteration. Thus, the computational complexity of each exploration mission can be measured by the number of executed explore actions until completion of the mission. Moreover, since we assumed that our robot is moving with a constant speed, then the overall completion time of the mission is equally proportional to the total distance traveled during it, i.e. the length of the travelled path until completion. Table I compares both quantities among the different testing environments presented earlier. We can see that the number of explore actions increases as the environment complexity increases. Similarly, the length of the travelled path is also affected by the tested environment layout. The presented
Fig. 8. Snapshots of the exploration in progress for a more complex environment resembling a residential apartment floor plan.

simulations are assumed to be 16m x16m (≈ 256m²), and as shown in the table the travelled distance is significantly smaller than the actual environments’ dimensions allowing the exploration mission to be accomplished in a relatively short time interval. After detailing the above experiments, we can say that our solution encapsulates a number of attractive characteristics:

1) Ease of Construction: The construction process is based on exploring $N_P$ edges in the DTT. This was enhanced by the proposed frontier selection method to accelerate exploration.

2) Compactness: The DTT structure requires a significantly smaller space than the traditional occupancy grid as shown in the analysis of the previously introduced experiments.

3) Completeness: The proposed exploration solution maps the environment geometrically allowing a more comprehensive representation.

4) Feasible Return Paths: The compactness of the DTT is significantly important for finding safe return paths even if the robot loses communication with the main station.

V. CONCLUSION AND FUTURE WORK

In this paper we have presented a new frontier-based exploration strategy. The proposed solution maps the environment geometrically and stores the map using the Dynamic Triangulation Tree structure DTT developed in this study. The geometrical information embedded in the DTT is used in the frontier selection step to reduce the exploration time by combining both the cost and gain values.

The work introduced in this article can be used for various extensions. During exploration the robot can locate detected targets or features on the constructed map. Given the connectivity of the DTT, it can be used to generate feasible paths for use by other robots or human agents in search and rescue missions. This approach is to be used in various applications such as indoor fire search and rescue for directing human personnel to victims locations and to produce a risk map of the detected fire sources. The proposed solution allows the use of robots with limited memory and computation requirements. It also minimizes the energy required since it is oriented toward minimizing the overall distance travelled during exploration.

REFERENCES


