A 2-Approximation Algorithm for the Online Tethered Coverage Problem

Gokarna Sharma*, Pavan Poudel†, Ayan Dutta†, Vala Zeinali†, Tala Talaei Khoei†, Jong-Hoon Kim*
*Department of Computer Science, Kent State University, Kent, OH 44242, USA
Emails: {sharma@cs.,ppoudel1@,vzeinali@,ttalaeik@,jkim@cs.}kent.edu
†School of Computing, University of North Florida, Jacksonville, FL 32224, USA
Email: a.dutta@unf.edu

Abstract—We consider the problem of covering a planar environment, possibly containing unknown obstacles, using a robot of square size $D \times D$ attached to a fixed point $S$ by a cable of finite length $L$. The environment is discretized into 4-connected grid cells with resolution proportional to the robot size. Starting at $S$, the task of the robot is to visit each cell in the environment that are not occupied by obstacles and return to $S$ with the cable fully retracted. Our goal is to minimize the total distance traveled by the robot to fully cover the unknown environment while avoiding tangling of the cable. In this paper, we present a novel online algorithm to solve this problem that achieves 2-approximation for the total distance traveled by the robot compared to the minimum distance that needs to be traveled. Our algorithm significantly improves the $2L/D$-approximation achieved by the best previously known online algorithm designed for this problem. The approximation bound is also validated using rigorous simulated experiments.

I. INTRODUCTION

Coverage path planning which requires a robot (or a team of robots) to completely cover a given area is a well-studied problem in robotics. It has many practical applications, such as autonomous sweeping, vacuum cleaning, and lawn mowing among others [7]. The goal is to plan path(s) so that the robot(s) can visit every point in the area. This problem has been studied heavily in the literature for a battery operated (i.e., cordless) robot assuming that the battery has an unlimited energy budget for moving long distances [2, 3, 6, 10, 16]. Therefore, given a robot, a single path can be planned to cover the given environment (possibly containing obstacles as well). Recently, this problem has been studied considering limited energy budget battery operated robots [17, 18].

However, there are applications which pose greater energy demands than an on-board battery can provide [5]. For example, an autonomous robot for urban street cleaning may require high power and can only be provided by AC power outlets [15]. In such situations, energy is provided to the robot using a cable connected to the power outlet and it is called a tethered or corded robot. Moreover, tethered-robot-based navigation is necessary in environments where information can only be transmitted via a communication cable, such as underground mines and electrostatic mine-fields [1, 15].

The planning of the path(s) to cover a given environment using a tethered robot is called the tethered coverage problem (denoted as the TC problem). The TC problem can be formally defined as follows: Given a robot of size $D \times D$ connected to a fixed base point $S$ (or outlet) by a cable of finite length $L$, the robot visits each point of the environment avoiding tangling the cable around obstacles present in the area [15]. TC is different from the untethered (or cordless) coverage problem (where there is no such cable attached to the robot; robot has an on-board battery). However, TC is a significantly complex problem since in addition to bypassing obstacles, the robot must avoid tangling and tearing its cable from its base point.

The offline version of TC (denoted as OFFLINE TC) assumes that the robot has the knowledge of the environment including obstacles (such as knowledge on locations, shape and size etc.) a priori. Shnaps and Rimon [15] presented an algorithm for OFFLINE TC. The online version of TC (denoted as ONLIENTC) assumes that the robot has no knowledge of the environment and the obstacles a priori. Shnaps and Rimon [15] presented an algorithm for ONLINE TC which obtains $2L/D$-approximation. The approximation is obtained comparing the length of the path(s) by the robot using the designed algorithm to the minimum length of the path(s) that need to be traversed by the robot (Definition 3).

Contributions. Similar to the model proposed by Shnaps and Rimon [15], we consider the online tethered path planning problem ONLINE TC with a robot of size $D \times D$ being connected to a fixed base point $S$ by a cable of finite length $L$. The cable is assumed to be a flexible non-stretchable cord, that can move freely within the obstacle-free portion of the environment. Following [15], the cable is assumed to be released by a spring loaded recoiling mechanism mounted on the robot (more on Section III), which keeps the cable taut at all times. The environment is discretized into cells (a discrete 4-connected grid) with size same to the robot size $D \times D$. The robot has sufficient on-board memory to store information necessary to facilitate coverage process. The goal is to find a set of paths $\Pi = \{\pi_1, \ldots, \pi_p\}, p \geq 1$, for the robot so that

- **Condition (a):** Each path $\pi_i$ (a sequence of cells) starts and ends at $S$,
- **Condition (b):** At the end of each path $\pi_i$, the cable is fully retracted, and
- **Condition (c):** The paths in $\Pi$ collectively cover the environment $P$, i.e., $\cup_{i=1}^{p} \pi_i = P$.

We will show that any algorithm satisfying simultaneously
conditions (a)–(c) correctly solves ONLINE TC. However, in this paper, we are interested in finding a set of paths that optimize the following performance metric:

- **Performance metric:** The total length of the paths in \( \Pi \), given by \( \sum_{i=1}^{P} |p_i| \), is minimized.

We establish the following main theorem.

**Theorem 1 (Main Result):** There is an algorithm that correctly solves ONLINE TC and guarantees a \( 2(1 - \frac{1}{N}) \)-approximation to the total length of the paths traversed by the robot compared to that of the minimum path length for the coverage; \( N \) is the total number of obstacle-free, accessible cells in \( P \) from \( S \) and \( D \) is the robot size.

The approximation achieved in Theorem 1 is a significant improvement compared to the \( 2L/D \)-approximation for ONLINE TC obtained by Shnaps and Rimon [15]. Our result also shows that the lower bound of \( 2 \log(L/D) \)-approximation given by Shnaps and Rimon [15] for risk-parameterized tethered coverage problem does not apply to ONLINE TC. Furthermore, we show that there are instances of ONLINE TC for which no algorithm achieves better than \( 2(1 - \frac{1}{N}) \)-approximation for the total length of the paths traversed (compared to the minimum length). This implies that our algorithm is optimal for ONLINE TC, in the worst-case. We have illustrated such an instance while proving the lower bound.

Our result is obtained using three techniques: (i) discretization of the environment, (ii) tree map construction, and (ii) a modified Depth First Search (DFS) traversal on the constructed tree keeping track of all new frontiers (cells) that are yet-to-be-visited by the robot. Each of these techniques is performed on-the-fly by the robot. The DFS traversal is constrained so that it respects the length \( L \) of the cable, meaning that it never reaches to a cell for which the distance from the fixed base point \( S \) is more than \( L \). The new frontiers are visited by the DFS traversal through usual forward and backtrack phases. We prove that when the new frontiers list becomes empty, then all the accessible cells in the environment are visited, which provides the complete coverage guarantee. We analyze the cost (length of the path) provided by our algorithm to show that it achieves the approximation of factor \( 2(1 - \frac{1}{N}) \). Extensive simulation results show that our proposed algorithm is fast, complete, and never tangles the cable.

## II. BACKGROUND

The earliest work on tethered navigation considered multiple robots attached by cables to different base points [9]. The objective was to make sure that each robot has its unique target and all the robots reach them simultaneously. Recently, a geometric approach is presented to search for the shortest tethered path along the configuration space for a mobile robot [12]. Tethered navigation with self-crossing of cable has been considered in [4, 19]. However, all these papers [4, 9, 12, 19] dealt with the offline path planning problem where the environment is known a priori and pre-processing can be done to optimize the paths of the robots. Our work in this paper handles the exploration issue in an unknown environment.

The efficiency of an online algorithm can be computed by its approximation [8, 11], which is (in the context of this paper) the upper bound on the ratio of the length of the path(s) by the candidate online algorithm to the minimum possible length of the path(s) to cover all the accessible cells in any environment. Shnaps and Rimon [15] recently presented a \( 2L/D \)-approximation algorithm to solve the ONLINE TC problem. They also provided a \( 2 \log(L/D) \)-approximation lower bound for the risk-parameterized online tethered coverage. In this paper, we significantly improve the approximation provided in [15] to \( 2 \). We have also found that the lower bound of [15] does not apply to ONLINE TC. This is notable since our approximation is independent of \( L \), whereas the approximation in [15] increases linearly with \( L \).

A closely related problem to TC is the untethered coverage problem. Icking et al. [11] and Gabriely and Rimon [6] studied this problem with no energy constraints, so that the robot can travel arbitrarily long distances without needing to recharge the battery. They established both lower and upper approximation ratio bounds. Recently, an energy-constrained version of the untethered coverage problem is studied in [14, 16, 17, 18]. Wei and Isler [17, 18] studied the offline version and Shnaps and Rimon [16] studied the online version with provable guarantees. On the other hand, Sharma et al. [14] has provided an optimal solution for this problem.

## III. PROBLEM MODEL AND PRELIMINARIES

We use the same model as in the previous work of Shnaps and Rimon [15]. We consider a robot \( r \) initially positioned at a fixed base point \( S \). The base point \( S \) is assumed to be inside a planar polygonal area \( P \). \( P \) may contain static planar polygonal obstacles. Following [15], the obstacles are assumed to be fixed in the sense that even when the cable is tangled around them, the force of the cable cannot move them from their positions. See Fig. 1 for an illustration of an example environment with different numbers of rectangular obstacles (\( O_i \)). \( P \) is discretized into square grids, meaning that the discretization structures the environment into a discrete 4-connected grid. The robot \( r \) has size \( D \times D \) so that it fits within a grid-cell in \( P \). The robot can move to any of the four neighbor cells of the cell that it is currently positioned. It has a cable of length \( L \) (that is attached to the base point \( S \)) once fully expanded; initially the cable is fully retracted and hence \( r \) can be at \( S \). Similar to [15], the cable is assumed to have a recoiling mechanism which keeps it taut at all times. With cable length of \( L \), the distance \( r \) can travel is at most \( L \) units far from \( S \), i.e., \( r \) can visit \( \lfloor L/D \rfloor \) cells in \( P \) of increasing distance from \( S \). Robot \( r \) is equipped with a compass (for the global coordinate system), a position sensor (e.g., GPS), and an obstacle-detection sensor (e.g., laser rangefinder). We presume that it can identify obstacles in any of its neighboring cells using the laser rangefinder. Moreover, we assume that initially \( r \) does not have any knowledge about \( P \), i.e., \( P \) is an unknown environment for \( r \).

We have the following observation on the size of \( P \). This is for the feasibility of covering all cells in \( P \).
Fig. 1: An illustration of cable’s stretching and recoiling mechanisms and tangling (crossing) of the cable.

**Observation 1**: If a cell of $P$ is located such that its distance from the fixed base point $S$ is $> [L/D] \cdot D$, then $r$ cannot fully cover $P$.

Since the cable length is $L$ when fully expanded, for any cell at distance $x$ from $S$, it needs to expand the cable by at most length $x \cdot D$ to reach there from $S$. What Observation 1 intuitively means is that any cell of $P$ cannot be farther than $[L/D]$ cells away from $S$ measured in $L_1$ distance (or Manhattan distance); otherwise some cells in $P$ will be left unexplored by $r$ since it cannot reach those cells. Therefore, the environment boundary is at most of radius $[L/D] \cdot D$ with its center at $S$. The list of cells that $r$ visits starting from $S$ constitute a path.

Notice that if there are some obstacles within $P$ located in such a way that they divide $P$ into two sub-polygons $P_1$ and $P_2$ with $P_1$ and $P_2$ sharing no common boundary, then $r$ cannot fully cover $P$. Therefore, we assume that there is no such cell $c$ in $P$. That means, there is (at least) a path from $S$ to any obstacle-free cell of $P$.

**Observation 2**: If there is at least a path connecting $S$ with any cell $c$ in $P$, then $c$ has at least one neighboring cell that is not occupied by any obstacle.

We call a cell free if it is not occupied by an obstacle. We call a cell reachable if it satisfies the definition below.

**Definition 1 (Reachable Cell)**: Any cell $c$ in $P$ is called reachable (or accessible) by the robot $r$, if and only if (a) it is a free cell, (b) it is within distance $[L/D] \cdot D$ from $S$ (Observation 1), and (c) there must be at least a path of consecutive free cells from $S$ to $c$ (Observation 2).

The online tethered coverage problem ONLINE TC can now be formally defined as follows.

**Definition 2 (ONLINE TC)**: Given an unknown polygonal environment $P$ possibly containing unknown obstacles and a robot $r$ having a cable of length $L$ attached to a fixed base point $S$ inside $P$, ONLINE TC is for $r$ to visit all the reachable cells in $P$ through a set of paths $\Pi$ such that

- Conditions (a)–(c) are satisfied, and
- The performance metric is minimized.

**Difficulties of the TC Problem.** One difficulty is the finding the best position to mount the cable’s recoiling mechanism. Shnaps and Rimon [15] argued that it would be better to mount it on the robot itself since friction aids in keeping the cable taut between successive contact points back to the base point. Moreover, the cable forms a single continuous curve from $S$ to the robot’s current position, and may not be stepped over by the robot. Additionally, TC must satisfy several constraints not typically encountered in untethered (cordless) coverage. First, the cable forms a dynamic obstacle that must be avoided by the robot. Second, the robot has to avoid crossing it, especially when circumnavigating an obstacle (Fig. 1). When a tethered robot happens to circumnavigate an obstacle, it must unwind the cable by retracing its path in order to complete coverage and eventually return to $S$. Third, the robot’s accessible area is restricted by the cable length. To achieve maximum coverage, the robot may have to reach peripheral points in the environment by using the shortest path from $S$ (Fig. 1).

**Approximation.** As mentioned earlier, we measure the quality of the online algorithm by its approximation ratio with the minimum cost possible for any algorithm. For $N$ accessible cells, the minimum cost is $N \cdot D$. Following Shnaps and Rimon [15], we compare the algorithm’s cost with $N \cdot D$ to obtain the approximation. There may be no algorithm that obtains cost $N \cdot D$ even with complete knowledge of $P$. Therefore, we also compare our algorithm’s performance (in terms of approximation, path length, and run time) against the best offline algorithm, denoted as OFFLINE ALG; essentially the algorithm of Shnaps and Rimon [15] for OFFLINE TC.

**Definition 3 (Approximation)**: An algorithm solving TC, $P_{TC}$, is a $k$-approximation algorithm, if the cost of solving any instance $p$ of $P_{TC}$ does not exceed $k$ times the minimum cost of solving $p$ knowing $p$ a priori.

### IV. Basic Results

We discuss two basic results below that lead to our proposed algorithm for ONLINE TC.

**Decomposition of the Environment.** Following [16, 17, 18], we decompose the environment $P$ into square cells of size $D \times D$, which is the size of the robot itself. Similar to [16, 17, 18], we assume that the obstacles are such that they do not partially occupy any cell in $P$, i.e., for a cell, an obstacle either occupies it completely or does not occupy it at all.

An *equi-distance contour* is a poly-line where the cells on it has the same distance to/from the base point $S$ (the left of Fig. 2). The cells on a contour can be ordered from one side to the other ($r$ is assumed to have the knowledge of the global coordinate system using the on-board compass). We use the left to right order. We can also order the contours based on their distance to $S$ in a strictly-increasing fashion.

Let $c$ be a cell and $C$ be contour. Let $d(c)$ denote the distance to $S$ from $c$ and let $d(C)$ denote the distance to $S$ from $C$. If $d(C_j) = d(C_i) + 1$, we say that contour $C_j$ is contour $C_i$’s next contour. The contour $C$ with $d(C) = 1$ is called the first contour; the first contour has at most 4 cells that are neighbors of $S$.

**Tree Map of the Environment.** Initially, the robot $r$ is placed at the fixed base point $S$. In this case, the tree, denoted by $T_P$, has only one node $S$, which we call the root of $T_P$. If there is no obstacle in $P$, each cell except the boundary cells of $P$ will have exactly four neighbor cells. Since $r$ has the global coordinate system, each of the (at most) four
neighboring cells can be consistently labeled West, North,
East, and South in the clockwise order starting from cell
in the West of \( r \). Note that this ordering does not affect
the algorithm as long as the ordering used in each cell is
consistent. That is, West in each cell should point the same
direction throughout the environment. This ordering helps us
in the DFS traversal process in the algorithm; otherwise we
cannot provide guarantee on the algorithm that all reachable
cells in the environment are covered by the algorithm.

Robot \( r \) picks the first free cell \( c_1 \) according to this order
and includes it in \( T_P \) as a child of \( S \). If the cell labeled West
is a reachable cell, then \( r \) picks that cell. Otherwise, it goes in
order of North, East, and South until it finds the first cell that
is reachable. Recall that there is at least one neighbor cell from
any cell that is not occupied by any obstacle (Observation 2).
We now have two nodes in \( T_P = \{S, c_1\} \), with \( c_1 \) as a child
node of \( S \). Furthermore, \( c_1 \) is a cell in the first contour \( C_1 \).

Since \( r \) is building \( T_P \) while exploring \( P \), it will move to
\( c_1 \) after it is included as a child in \( T_P \). The robot \( r \) then again
repeats the process of building \( T_P \) from its current cell \( c_1 \).
While at \( c_1 \), \( r \) is only allowed to add one of the neighboring
cells of \( c_1 \) that are in the second contour \( C_2 \) (i.e., \( d(C_2) = 2 \))
as a child of \( c_1 \). For this, \( r \) will include a neighboring cell \( c_2 \)
of \( c_1 \) in \( T_P \) only if \( c_2 \) is a cell in contour \( C_2 \). Furthermore,
if some cell is already a part of \( T_P \), then this cell will not
be included in \( T_P \) again. This process will then continue. The
right of Fig. 2 provides an illustration of the tree map \( T_P \)
developed for the environment \( P \) shown on the left.

Essentially, any edge of \( T_P \) connects two cells \( c_i, c_{i+1} \) of \( P \)
such that \( c_i \in C_i \) and \( c_{i+1} \in C_{i+1} \), for \( 0 \leq i \leq \lfloor |L/D| \rfloor - 1 \); In Fig. 2, each cell of contour \( C_i \) on the environment \( P \)
on the left are at depth \( i \) in \( T_P \) shown on the right. Therefore,
using this approach, all the cells in the first contour \( C_1 \) will be
children of \( S \) (the root of \( T_P \)), all the cells in the second
contour \( C_2 \) will be children of the nodes of \( T_P \) that are cells
in the first contour \( C_1 \), and so on. Let \( T_{P;\text{final}} \) be the tree
map of the environment \( P \) after all the reachable cells of \( P \)
are included in \( T_P \). Let \( \text{Depth}(T_{P;\text{final}}) \) denote the depth of
the tree \( T_{P;\text{final}} \); the root \( S \) has depth 0, the child node of \( S \)
has depth 1, and so on. Let \( N_x \) be the nodes of \( T_P \) such that
the distance from \( S \) to each node in \( N_x \) is \( x \).

**Lemma 1:** In tree \( T_{P;\text{final}} \), each cell \( e \in C_x \) is positioned
at depth \( x \). Furthermore, any non-leaf node of \( T_{P;\text{final}} \) has at
least one and at most four children nodes.

**Proof:** Follows easily from construction of \( T_P \) since the
cells at contour \( C_x \) become the children of the cells of contour
\( C_p \) such that \( p = x - 1 \), for \( 1 \leq x \leq \lfloor |L/D| \rfloor \).

**Lemma 2:** For complete coverage, the corresponding tree
map \( (T_{P;\text{final}}) \) of any unknown polygonal environment \( P \),
will have \( \text{Depth}(T_{P;\text{final}}) \leq \lfloor |L/D| \rfloor \).

**Proof:** This lemma follows combining the results of
Observation 2 and Lemma 1.

A \( 2(1 - 1/N) \)-Approximation Lower Bound. Suppose the
robot \( r \) has the cable of fixed
length \( L \). Let \( P \) be a planar environment of size \( L/2 \times L/2 \).
Let the robot be of size \( D \times D \) and \( L > D \). Let the environment
\( P \) is such that a single square obstacle of size \( (L/2) - 2D \times
(L/2) - 2D \) is positioned inside \( P \) leaving only the corridor of
width \( D \) around the boundary of
\( P \) (see Fig. 3). According to this construction, there will be
exactly \( N = (4L/2D) - 4 \) cells of size \( D \times D \) available
for the robot \( r \) to cover. Let the base point for the robot \( r \)
be the bottom left corner cell of \( P \). The approximation is
obtained comparing the cost of the best possible algorithm
with minimum cost \( N \cdot D \) (Definition 3).

**Theorem 2 (Lower Bound):** For the construction above,
your algorithm for OnlineTC has \( 2(1 - 1/N) \)-approximation
to fully cover \( P \).

**Proof:** Following Fig. 3, the cost is at least \( 2(N - 1)D \).
Therefore, there exists a lower bound of \( 2(N - 1)D/(N \cdot D) \geq
2(1 - 1/N) \)-approximation to cover \( N \) cells in \( P \).

V. ALGORITHM

We now present an algorithm that achieves a \( 2(1 - 1/N) \)-
approximation (analysis is in Section VI) for OnlineTC in
an unknown planar environment \( P \). The pseudocode is given
in Algorithm 1.

A. Overview of the Algorithm

The main idea is to incrementally explore the environment
\( P \) while simultaneously constructing a tree map \( T_P \) to keep
track of the new frontiers that still need to be covered. Initially,
\( r \) is placed at the fixed base point \( S \) with the cable fully
retracted. Robot \( r \) then proceeds to cover \( P \) by using Depth
First Search (DFS) traversal [13]. Robot \( r \) starts its DFS
traversal from \( S \) and visits the leftmost cells of each contour
with increasing depth until it reaches contour at depth \( |L/D| \)
from \( S \), i.e., in contour \( C_{\lfloor L/D \rfloor} \). In some cases, \( r \) does not need
to reach contour \( C_{\lfloor L/D \rfloor} \), for example, when the obstacles
are such that there is no free cell beyond some contour
\( C_{D'} \), \( D' < |L/D| \).

Consider the situation of \( r \) after reaching to a cell in a contour
\( C_{D'} \). If \( r \) cannot continue visiting a cell in the next
contour \( C_{D'+1} \), it will backtrack to visit the new frontiers
(not-yet-visited) of \( C_1 \) to \( C_{D'-1} \). The process of visiting
the leftmost previously unvisited cell in each contour of increasing distance is called the forward phase and the process of retreating back from a cell from which no forward phase is possible, is called the backtrack phase. The forward and backtrack phases are executed by \( r \) alternatively until all the cells in \( P \) are visited. Initially (before the algorithm starts), every cell of \( P \) is unvisited. The tree construction \( T_P \) helps to keep track of whether there are cells that still need to be visited by \( r \) and also to control the total distance traveled for visiting all the reachable cells in \( P \). Without this, the DFS traversal cannot provide both coverage and approximation guarantees.

While covering the cells using DFS traversal, the robot \( r \) keeps track of its current distance \( D_{\text{cur}} \) from \( S \). \( D_{\text{cur}} \) is depth of the cell in \( T_P \) where \( r \) is currently positioned. This distance helps in backtracking while the depth tree reaches \( \lceil L/D \rceil \).

**Algorithm 1: ONLINE\textsc{TCalg}**

1. **Robot:** Initially at the base point \( S \) and knows \( L \) and \( D \).
2. **Environment:** Planar area with radius \( L \) with center \( S \) possibly containing obstacles; obstacle positions and numbers not known.
3. **Sensors:** Compass, position sensor, and obstacle-detection sensors capable of locating current position and detecting obstacles in the cells adjacent to the robot’s current cell.
4. **Data Structures:** Tree map \( T_P \) and new frontier stack \( F \).
5. **Initialize:** \( T_P = \{ S \} \), \( F = \{ S \} \), distance (depth) \( D_{\text{cur}} = 0 \).
6. while \( F \neq \emptyset \) do
7. \( C_x \leftarrow \) contour of cell \( x \) (depth of \( x \) from \( S \) in \( T_P \));
8. \( p \leftarrow \) top element in \( F \);
9. remove \( p \) from \( F \);
10. \( p_i \leftarrow \) leftmost neighboring cell of \( p \) among at most 4 neighboring cells of \( p \) not occupied by any obstacle;
11. insert the second, third, and fourth neighboring cells (if any, in the order North, East and South) of \( p \) in \( F \);
12. \( p_i \leftarrow \emptyset \);
13. if \( p_i \neq \emptyset \) then
14. if \( d(C_{p_i}) > D_{\text{cur}} \) then \( p_i \leftarrow p_i \);
15. else if there is an obstacle in the neighboring cell of \( p_i \) in the direction of the base point \( S \) or \( C_{p_i} > C_p \) then
16. \( d(C_{p_i}) \leftarrow D_{\text{cur}} + 1 \); \( p_i \leftarrow p_i \);
17. if \( p_i \neq \emptyset \) then
18. move to \( p_i \) from \( p \);
19. insert \( p_i \) in \( T_P \) making \( p_i \) a child node of \( p \);
20. \( D_{\text{cur}} \leftarrow D_{\text{cur}} + 1 \);
21. if \( p_i \neq \emptyset \) and \( p_i \) has unvisited neighboring cells in \( F \) and \( D_{\text{cur}} < \lceil L/D \rceil \) then
22. pick the leftmost unvisited cell and continue the DFS traversal;
23. else
24. backtrack towards \( S \) following the path in \( T_P \) to reach the first node (or cell) which has unvisited neighboring cells in \( F \) and for each backtrack traversal of an edge in \( T_P \), set \( D_{\text{cur}} \leftarrow D_{\text{cur}} - 1 \);

**B. Detailed Description of the Algorithm.**

We now describe Algorithm 1 in detail. We call our algorithm ONLINE\textsc{TCalg}. It uses two data structures: (1) The tree map \( T_P \) of the known part of the unknown environment \( P \) and (2) A new frontier stack \( F \). The tree map \( T_P \) at any time holds the cells already covered by ONLINE\textsc{TCalg} as well as their reachable unvisited neighbors. The new frontier stack \( F \) consists of the unvisited reachable cells in \( T_P \) (that are neighbors of the cells already in \( T_P \)). \( F \) also holds the distance value \( D_{\text{cur}} \leq \lceil L/D \rceil \), which is the depth of the cells in \( T_P \).

The new frontier list \( F \) is a stack of cells. Since \( F \) is a stack, the cells with higher distance from \( S \) in \( T_P \) are on the top of the stack and will be popped first. Robot \( r \) visits a cell \( c_i \) when \( d(C_{c_i}) > D_{\text{cur}} \) in \( T_P \).

When the DFS traversal cannot go further in its forward phase, that means either it reached a cell at distance \( \lceil L/D \rceil \cdot D \) or there is no next cell to visit (the cell at next contour is occupied by an obstacle). In this case, \( r \) backtracks to reach the first node that has unvisited neighbor cells in stack \( F \). The traversal finishes when \( F = \emptyset \). If there are \( N \) reachable cells in \( P \), then we will show that when \( F = \emptyset \) then \( r \) has already visited all \( N \) cells and hence, ONLINE\textsc{TCalg} is achieved.

**Execution Examples of ONLINE\textsc{TCalg}**. Fig. 4 illustrates how ONLINE\textsc{TCalg} covers \( P \). We have two scenarios with obstacles \( O_1 \), \( O_2 \) in left one and a single obstacle \( O_3 \) in the right one. The robot \( r \) starts the DFS traversal from \( S \). The paths colored blue denote the forward traversal phase of ONLINE\textsc{TCalg} and the paths colored red denote the backtrack traversal phase in both scenarios.

In the left of Fig. 4, \( r \) performs the forward phase of DFS traversal in the increasing order of shortest \( L_1 \) distance from \( S \) to each of the cell (which is essentially the contour number of the cell) in the tree map \( T_P \). The contour numbers of the cells behind the obstacles \( O_1 \) and \( O_2 \) w.r.t. \( S \) are in increasing order going from one end to the other end of each obstacle and hence the robot \( r \) can follow the same increasing order path during the traversal.

![Fig. 4: Two execution runs of ONLINE\textsc{TCalg.}](image1)

In the right of Fig. 4, the contour numbers of the cells

![Fig. 5: The DFS traversal and the corresponding tree map.](image2)
behind the obstacle w.r.t. $S$ are first decreasing and then in increasing order after crossing the horizontal line through $S$ when traversing alongside the obstacle from top or bottom. This is because, in the left of Fig. 4, the horizontal and vertical lines passing through $S$ do not cross any obstacle and in the right of Fig. 4 the horizontal line passing through $S$ crosses the obstacle $O_3$.

Note that robot $r$ cannot perform the forward phase of DFS traversal in decreasing order of contour numbers. Algorithm 1 handles this problem by updating the contour numbers of each cell behind the obstacle w.r.t. $S$ in increasing order on-the-fly (line 15). We discuss this approach below in Section V-C. Robot $r$ then performs the DFS traversal. This traversal is depicted in more details in the left of Fig. 5. The corresponding tree map built on-the-fly is shown in the right of Fig. 5.

In Figs. 4 and 5, the paths depict the robot moving in North, East, and South directions. The robot moving North happens when from its current cell, there is no cell on the West of it to visit but there is a reachable cell on North of it. The robot moving East happens when from the current cell, no cell to move to West and then North is possible. Similarly, the robot moving South happens when there is no cell to move to West, North, or East is possible from the current cell.

C. A Corner Case and Its Solution

Consider the situation of obstacle position and fixed base point position as depicted in the right of Fig. 4. What is happening here is the horizontal line passing through $S$ is crossing obstacle $O_3$. This kind of scenario is the most difficult to handle in our algorithm. This is because providing the contour numbers to the cells in the right of $O_3$ based on their distances from $S$ creates the problem that $r$ may not visit those cells using Algorithm 1. This is because the contour numbers for those cells are smaller than the contour number of the cells on North, South, and West of the obstacle. For example, see the left of Fig. 6. Algorithm 1 requires the robot to visit the cells in an increasing order of contour numbers. Therefore, without careful update of the contour numbers of the affected cells, some portion of the environment might remain unvisited.

We solve this problem in Algorithm 1 using an approach where the robot $r$ detects this problem and changes the contour numbers of those cells on-the-fly. See for example the right of Fig. 6 that depicts how the contour numbers for the cells on the right of $O_1$ in the left of Fig. 6 are updated on-the-fly by the robot.

The approach of changing the contour numbers for a cell $c_j$ is as follows. Let the robot $r$ be currently at cell $c_i$ with depth $D_i$. Let $c_j$ be the neighboring cell of $c_i$ with depth $D_{i-1}$. Let $L_j$ be a line connecting $c_j$ with $S$. The robot $r$ checks if $L_j$ crosses an obstacle. If this is true, $r$ checks whether the neighboring cell of $c_j$ on line $L_j$ is occupied by an obstacle. If this is also true, then $r$ updates the contour number of $c_j$ to $D_i + 1$. See for example the left of Fig. 6. Suppose $r$ is currently at the middle cell in contour $C_8$. Consider cell 7 just below 8. If we draw $L_7$ connecting $S$ with 7, then it crosses obstacle $O_1$. From the current cell of $r$, it can sense that neighboring cell of 7 on $L_7$ is occupied by $O_1$. In this case, $r$ changes the contour number of cell 7 to 9 (which is $r$’s contour number 8 plus 1). This then affects all the cells on the right of 7 as well and they will also be updated accordingly. For example, the contour number 8 on South of contour 9 (and East of newly updated cell 9) becomes 10. The complete depiction of the changes is given in the right of Fig. 6.

This case may appear when a vertical line and/or horizontal line passing through $S$ crosses an obstacle (or separate obstacles) and there is a path for a robot to reach the cells behind the obstacle by circumnavigating the obstacle. In this case, the contour numbers of the cells behind the obstacle w.r.t. $S$ are in decreasing order going from one end to the other. The robot then uses our approach discussed above and fixes the contour numbers on-the-fly so that all reachable cells beyond the obstacle(s) are covered.

VI. ANALYSIS OF THE ALGORITHM

A. Complete Coverage Guarantee of ONLINETCALG

The goal here is to show that using ONLINETCALG, the robot $r$ that is initially at the fixed base point $S$ with the cable fully retracted covers all reachable grid cells of the environment $P$ and returns to $S$ with cable fully retracted.

Lemma 3 (No Tangling): The robot $r$ never crosses (or tangles) its cable during the execution of ONLINETCALG.

Proof: The full proof is omitted due to space constraints. However, it is quite evident that using the tree map $T_P$ (which guarantees no cycle) and DFS traversal on $T_P$ with the depth constraint of $\lfloor L/D \rfloor$, the robot never runs into a situation where the cable is tangled. ■

Lemma 4 (Coverage Guarantee): Using algorithm ONLINETCALG, the robot $r$ completely covers the environment.

Proof: Let us assume that ONLINETCALG does not provide full coverage of the unknown environment $P$ and thus, stops before some reachable cells of $P$ still to be inserted into $T_P$. Let $c_1$ be the cell of contour $C_x$ such that $C_x$ is the smallest contour among all other cells of $P$ that are reachable but uncovered by $r$. Since a path of length at most $L$ exists from $S$ to $c_1$, $x \leq \lfloor L/D \rfloor$ (depth in $T_P$). Let $c_2$ be the cell adjacent to $c_1$ along the path from $c_1$ back to $S$. Since $d(C_{c_2}) < d(C_{c_1})$ and $c_2$ holds the minimal $C$ value among all reachable uncovered cells, $c_2$ must be a covered cell in $P$. ONLINETCALG inserts every reachable neighbor (at most $4$) of the current cell in the list of new frontiers yet to be visited (line 11, Algorithm 1) Therefore $c_1$ must have been inserted into $F$ when the robot explored $c_2$ and eventually covered
by \textsc{OnlineTcalg}. Therefore, \textsc{OnlineTcalg} covers every cell reachable from $S$ by a cable of length $L$.

\subsection*{B. Approximation of the Algorithm}

We establish in this section an upper bound on the path length generated by \textsc{OnlineTcalg} to cover all the cells of $P$ reachable from $S$. We use the notion of approximation given in Definition 3 for this purpose. We assume that the robot $r$ travels with uniform velocity. Finally, we measure \textsc{OnlineTcalg}'s total path length (denoted as $\textsc{Alg}$) relative to the minimum path length (denoted as $\textsc{Min}$). We first establish a lower bound on $\textsc{Min}$ in any planar polygonal environment $P$. Denote by $N$ the total number $(D \times D)$ of cells in $P$ that are reachable by a cable of length $L$ from $S$.

\textbf{Lemma 5 (Minimum Cost):} For any planar polygonal environment $P$ consisting of $N$ cells reachable by a cable of length $L$ from a fixed base point $S$, $\textsc{Min} \geq N \cdot D$.

\textbf{Proof:} The purpose in TC is to cover all reachable cells. For $N$ cells reachable by a cable of length $L$ from $S$, any shortest path for the robot $r$ to visit those $N$ cells must have length $N$ times the distance $D$ the robot $r$ needs to traverse from one cell to one of the four cells adjacent to it. Hence, $\textsc{Min} \geq N \cdot D$.

We now prove the upper bound on $\textsc{Alg}$ for \textsc{OnlineTcalg}. We start with the following lemma.

\textbf{Lemma 6:} Using Algorithm \textsc{OnlineTcalg}, the robot $r$ traverses any single edge of $T_{P,final}$ at most 2 times.

\textbf{Proof:} Let $e_{12} \in T_P$ be an edge that connects a parent $(v_1)$ and a child $(v_2)$ node in $T_P$. The edge $e_{12}$ is traversed by the robot first time when it is in the forward phase and then again when it is in the backtrack phase. We show here that if $e_{12}$ is already visited for the second time in the backtrack phase, it will then never be visited again during the whole execution of the algorithm. Let $T^w_p$ be a sub-tree of $T_P$ with the root node $v_2$. We can say that the edge $e_{12}$ is the only edge that connects $T^w_p$ with the rest of the tree $T_P \setminus T^w_p$. Therefore, once $r$ backtracks to $v_1$ from $v_2$, it will never visit $T^w_p$ again and hence the edge $e_{12}$ will never be used.

\textbf{Lemma 7 (Cost Upper Bound):} For any planar polygonal environment $P$ consisting of $N$ cells (of size $D \times D$) reachable by a cable of length $L$ from a fixed base point $S$, Algorithm \textsc{OnlineTcalg} has cost $\textsc{Alg} \leq 2(N - 1) \cdot D$.

\textbf{Proof:} We know that any tree with $N$ nodes has exactly $(N - 1)$ edges and $T_{P,final}$ that is constructed by \textsc{OnlineTcalg} while under coverage is clearly a tree. Moreover, we have from Lemma 6 that each edge of $T_{P,final}$ is traversed by the robot $r$ at most 2 times during the execution of \textsc{OnlineTcalg}. Since the edges of $T_{P,final}$ denote a transition from one cell to one of its four adjacent cell, the length the robot $r$ needs to move while traversing an edge of $T_{P,final}$ is $D$. Hence, combining all these results, we have $\textsc{Alg} \leq 2(N - 1) \cdot D$.

\textbf{Proof of Theorem 1:} Combining the results of Lemmas 5 and 7, we have that $\textsc{Alg}/\textsc{Min} \leq 2(N - 1)D/ND \leq 2(1 - 1/N)$.

\section*{VII. Evaluation}

In this section, we present the results obtained by performing simulation experiments with our algorithm using Python. The simulation is performed on an Intel Core i7-7700K processor and 32 GB RAM. We have varied the size of the environment $P$ from $20 \times 20$ cells to $400 \times 400$ cells, as well as position of base point $S$ within $P$, and shapes, sizes and positions of the obstacles unknown to $r$. We have used a robot of size $2 \times 2$ (i.e., $D = 2$). The length $L$ of the cable is set to $2D$ times the side of the environment. We have considered 5 different environment configurations of varying sizes. We classify these configurations below (shown in Fig. 9):

- (conf 1) There is no obstacle in $P$ and the base point $S$ is located at a random cell in $P$.
- (conf 2) $S$ is located in a boundary cell of $P$, a large obstacle is placed inside $P$ leaving the corridor of width $D, 2D, 3D, 4D$ etc. around the boundary of $P$.
- (conf 3) $S$ is located at the center of $P$, four obstacles are placed inside $P$ leaving only the corridor of width $D, 2D, 3D, 4D$ etc. between the obstacles and $P$'s boundary.
- (conf 4) Concave obstacles are placed within $P$.
- (conf 5) Concave obstacles are placed in such a way that either a horizontal or vertical line passing through $S$ crosses at least one obstacle.

Each test case is run 10 times and the average result is presented here. First we show the path lengths incurred by
Fig. 9: An illustration of five different configurations (conf 1 to conf 5) and actual paths followed by \( r \) using ONLINE\textsc{TCALG}. In this illustration, environment size is set to \( 20 \times 20 \).

different algorithms including our proposed ONLINE\textsc{TCALG} (Fig. 7). \( MIN = N \cdot D \) is the minimum possible cost for covering \( N \) cells with \( D \) being the cell size. We also compare the path lengths obtained by ONLINE\textsc{TCALG} against an optimal offline algorithm similar to [15]. The plots in Fig. 7 show that our proposed ONLINE\textsc{TCALG} performs comparably with the OFFLINE\textsc{ALG}. However, as expected, both of their path costs are higher than \( MIN \). On the other hand, ONLINE\textsc{TCALG} achieves significantly lower path costs compared to the \( 2L/D \)-approximation provided in [15].

The plots in Fig. 8 validate the theoretical worst-case approximation provided by ONLINE\textsc{TCALG}. We can observe that the approximation of ONLINE\textsc{TCALG} is always \( \leq 2 \). On the other hand, even with a priori knowledge about the environment, the OFFLINE\textsc{ALG} does not guarantee the optimal path(s) in all types of environments validating our lower bound approximation (Section IV).

VIII. CONCLUDING REMARKS

We have presented a novel algorithm for ONLINE\textsc{TC} and showed that the cost approximation of our proposed algorithm is always within a factor of 2 compared to the minimum cost possible for complete coverage of an unknown environment. Thus we significantly improve the \( 2L/D \)-approximation of the best previously known algorithm for solving ONLINE\textsc{TC}. The simulation results show the effectiveness of our algorithm. An immediate direction of future work would be to investigate whether we could generalize this result to continuous robot motion.
REFERENCES


