Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 Keller Hall
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 18-004

Coverage path planning under the energy constraint

Volkan Isler, Minghan Wei

February 16, 2018

Revised

Coverage Path Planning Under the Energy Constraint

Minghan Wei and Volkan Isler

***Abstract*— In coverage applications, a common assumption is that the robot can fully cover the environment without recharging. However, in reality most mobile robot systems operate under battery limitations. To incorporate this constraint, we consider the problem when the working environment is large and the robot needs to recharge multiple times to fully cover the environment.**

We focus on a geometric version where the environment is represented as a polygonal grid with a single charging station. Energy consumption throughout the environment is assumed to be uniform and proportional to the distance traveled. We first present a constant-factor approximation algorithm for contour-connected environments. We then extend the algorithm for general environments. We also validate the results in experiments performed with an aerial robot.

I. INTRODUCTION

Coverage is a fundamental robotics problem. In many practical coverage applications, energy limitations would prevent a robot from covering the entire environment in one iteration. Before the battery runs out, the robot has to visit a charging station to get fully recharged to continue its work. As a result, instead of a single path, we may need to plan multiple paths to cover the environment. Each path should start and end at a charging station. The robot's energy budget should be enough to follow each path without recharging. In this paper, we study the coverage path planning problem with this energy constraint.

The literature on coverage problem can be divided into two groups based on the environment representation. (i) The environment is represented as a graph. Without the energy constraint, the coverage problem becomes the well-known Traveling Salesperson Problem (TSP). The Vehicle Routing Problem (VRP) considers the energy and capacity constraints when visiting vertices. One version is called Distance Vehicle Routing Problem (DVRP), which models the energy consumption proportional to the distance traveled. The VRP is mainly solved by integer programming and heuristic methods [9]. Nagarajan considered DVRP on tree metrics and proposed a 2-approximation algorithm [12]; (ii) The second common representation of the environment is a polygon. The robot is represented as a unit disk or square. The goal is to move the robot in the polygon so that every point in the polygon is covered. The choice of the representation depends on the practical application.

Coverage path planning problem without energy constraints has been studied extensively in the literature [7]. Well-known coverage strategies include the boustrophedon decomposition coverage with the back-and-forth motion [4], [11], spiral path coverage [8], and spanning-tree based coverage [6]. The boustrophedon coverage and spiral path coverage can be adapted to perform in an online fashion by tracing the uncovered area [14] [3]. The objective of these problems is to completely cover the environment. Yoav and Elon restrict the robot to move rectilinearly, the algorithm is optimal when there is no zero-thickness in the spanning tree [6].

Coverage with energy constraints is a relatively new topic. Shnaps and Rimon [13] studied this problem and modeled the energy cost by the path length. They provided an approximation algorithm with a factor of $\frac{1}{1-\rho}$, where ρ is the ratio between the distance of the furthest cell from recharging station and half of the energy budget. This factor can be arbitrarily large when ρ approaches 1.

Our contributions: We revisit the setting in Shnaps and Rimon [13] and present a constant-factor approximation algorithm for the energy-constrained coverage problem for *equi-distance-contour-connected* environment (contour-connected for short) when the robot is restricted to axis-parallel motion. We then generalize this algorithm to arbitrary polygons by partitioning a given polygon into contour-connected pieces. Compared with prior work, the deviation of the cost of our algorithm from the optimal cost remains bounded. We also present the results from a field experiment performed with an autonomous aerial vehicle.

II. PROBLEM FORMULATION

We are given an environment represented as a unit-grid laid out on a polygon P. P has a single charging station S in it. The robot is represented as a unit square which can only move rectilinearly in P. The robot's energy consumption is assumed to be proportional to the distance traveled. We use 'path length' and 'energy cost' interchangeably. Due to the energy constraint the robot can only move at most B units after a full charge. Then it has to get recharged at S. The goal is to find a set of paths, $\Pi = \{\pi_1, \pi_2, ..., \pi_n\}$, for the robot such that $\bigcup_{i=1}^{n} \pi_i = P$ and the number of paths in Π is minimized. In addition, each path should start and end at S and $|\pi_i| \leq B$ for $i = 1, 2, ..., n$.

A related goal is to minimize the total length of the paths in Π. In our analysis, we bound the number of paths first, then bound the total length.

III. A GENERAL TSP BASED ALGORITHM

Before we present our constant-factor approximation algorithm, we first show how TSP-partitioning techniques (see, e.g. [2], [5]) can be used to obtain a simple algorithm which matches the performance of [13].

[1] The authors are with the Computer Science and Engineering Department at the University of Minnesota. Emails: {weixx526,isler}@umn.edu

Let r be the radius of the environment (the furthest distance from S). We must have $B = 2r + \alpha$ with $\alpha \geq 0$ because otherwise the robot does not have enough battery to reach the furthest cells. Let T be the optimal TSP tour of the grid cells (or vertices), and OPT be the optimal number of battery-constrained paths to cover the environment. We have $OPT \geq T/B$.

We now partition T into $k = T/(B-2r)$ segments (when $\alpha > 0$) and form subtours by connecting the beginning and the end of each segment to S. Each subtour has length at most B. The ratio of the total number of subtours to the optimal tour is at most

$$\frac{T/(B-2r)}{T/B} = 1 + \frac{2r}{\alpha}$$

A few remarks are in order: Since TSP is NP-Complete, we can not compute the optimal tour T. However, in Euclidean settings, we can use a PTAS to obtain an arbitrarily close solution [1]. Second, this argument can apply to general environments, e.g. with non-uniform costs as long as the environment can be represented as a graph. Furthermore, if the robot's energy budget is large enough, e.g. $B > 3r$, this yields a 3-approximation algorithm. However, in theory α can be arbitrarily small. Just like the algorithm in [13] [10], the theoretical performance of this technique can be arbitrarily bad.

In the rest of the paper, we deal with this issue and show how a depth-first-search-like coverage strategy yields a 4-approximation algorithm in certain environments.

IV. Preliminaries

In this section, we present the definition of contour-connected environments and introduce several terms and notations which are necessary for our analysis.

An *equi-distance contour* (contour for short) is a polyline the cells on which have the same distance to the charging station. A cell is called a *split cell* if this cell is adjacent to the boundary and the equi-distance contour splits into two segments at this cell, as defined by Shnaps and Rimon [13]. The *contour-connected* environment refers to an area without split cells. Note that the cells on the outer most contours are not considered to be split cells though they are adjacent to the boundary. Any convex environment with a charging station at an acute vertex is contour-connected. Fig. 1 shows three examples, two of which are contour-connected and one is not. Second and third examples illustrate that the position of the charging station S affects whether the environment is contour-connected or not.

When covering the environment, if any cell on a contour C is visited multiple times, we call C a *saturated* contour. $\bar{C}$ denotes the area further than C to S. For two cells p and q, if any shortest path from S to q passes through p, we say q is *accessible* from p. Let C_{sub} be a subsegment of C. Then subarea $\bar{C}_{sub}$ is the union of all the cells in $\bar{C}$ which are accessible from the cells on C_{sub}, as shown in Fig. 2. We use $d(\cdot)$ to denote the distance from the contour or the cell

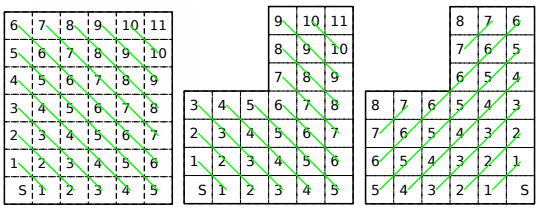


Fig. 1. From left to right, (1)convex contour-connected. (2) non-convex contour-connected. (3) non-contour-connected. The value in each cell is its $L1$ distance to the charging station S.

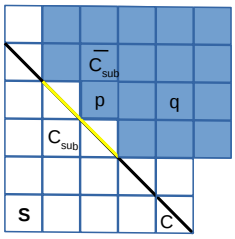


Fig. 2. An example of C_{sub} and $\bar{C}_{sub}$. C_{sub} is colored yellow and $\bar{C}_{sub}$ is colored blue.

to S, and use $|\cdot|$ to denote the number of the cells on the contour or in the area.

A contour-connected environment has the following properties. The proofs are presented in the appendix.

Proposition 1: If C is an equi-distance contour in a contour-connected environment, then C is a line segment.

Proposition 2: Let C and C' be two adjacent equi-distance contours. Then $||C| - |C'|| \leq 1$.

Proposition 3: Let C and C' be two contours in a contour-connected environment. Suppose C' is further than C. $c_1, c_2, ..., c_p$ are the cells on C from one side to the other. $c'_1, c'_2, ..., c'_q$ are the cells on C' from the same side to the other. (c_i is defined to be c_p if $i > p$, and c'_i is defined to be c'_q if $i > q$). Then c'_i is accessible from c_i for all i.

V. Coverage Strategy for Contour-connected Environments

We now present our coverage algorithm for contour-connected environments and prove its performance. Let r be the robot's position. We define three types of motions for the robot. (i) $Advance$: For all the unvisited cells c with $d(c) > d(r)$, the robot moves to the closest one along the shortest path. (ii) $Follow$: the robot follows the current contour first, then follow the closer uncovered contours to S. (iii) $Retreat$: The robot returns to S along the shortest path. Note that the $Advance$ motion may contain the $Follow$ motion. Initially r=S. Re denotes the current remaining energy budget of the robot.

Algorithm 1 proceeds as follows. The robot first goes to the furthest uncovered cell in the environment using the $Advance$ motion. Then it follows the contours by the $Follow$ motion. When the remaining energy is just enough for going back to S, the robot retreats. For the $Advance$ motion, there may be multiple candidate cells to move to. In

Algorithm 1 Coverage of contour-connected environments.

Output: Π_{sol}.

1: $\Pi_{sol} = \emptyset$; i=1;
2: Start recording path π_i;
3: **while** P is not fully covered **do**
4: $Advance$ to the furthest uncovered cells.
5: $Follow$ until $Re = d(r)$.
6: **if** i is odd **then**
7: record current position, p=r.
8: **end if**
9: $Retreat$. Recharge the robot; Add π_i to Π_{sol};
10: i++; Start recording path π_i;
11: **if** i is even **then**
12: Move to p along shortest path;
13: **end if**
14: **end while**

this case, we choose the one adjacent to the boundary or the previous path to avoid splitting the environment. The strategy is presented in Algorithm 1, which essentially performs a depth-first-search-like coverage. An execution example is shown in Fig. 3.

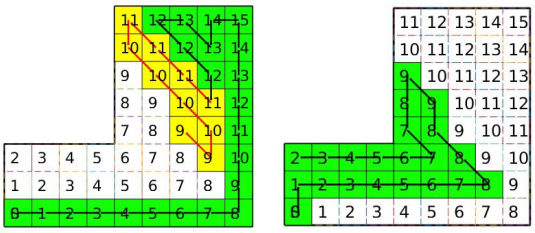


Fig. 3. An execution of Algorithm 1. For simplicity, we use a straight line for the $Follow$ motion and do not show the $Retreat$ motion. The second path starts from where the first path retreats and is colored yellow.

To prove the performance of the algorithm, we need to compare Π_{sol} to the optimal solution. We take an indirect approach. We first transform the optimal solution into another feasible solution in the following way. We split each path in the optimal solution into two paths from its midpoint. For both parts, the robot starts from S and follows the sub-paths to the midpoint. Then it retreats to S along the same trajectory. Thus the retreating phase does not visit new cells. Fig. 4 shows such an example. We get a path set, Π^*_{split}, which fully covers the environment and has twice the number of the paths of the optimal solution. We compare Π_{sol} and Π^*_{split} to bound the number of paths in our solution.

We now present an overview of the analysis. We first show that when the robot reaches any saturated contour C the first time by following paths in Π_{sol}, it still has $B - d(C)$ energy to visit at least $\frac{B}{2} - d(C)$ cells before retreating. But when the robot reaches C the first time by following the paths in Π^*_{split}, it can visit at most $\frac{B}{2} - d(C)$ cells (Lemma 2). Then we show that if Π_{sol} needs n paths to cover $\bar{C}$, Π^*_{split} needs at least n paths to cover $\bar{C}$. Then we bound the path number.

In our analysis, we do not consider the cells covered by the

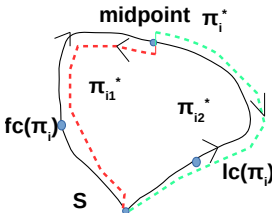
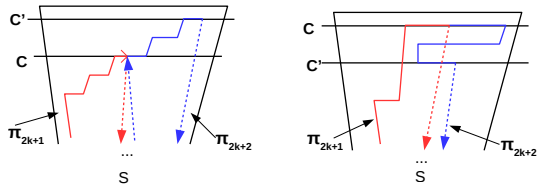


Fig. 4. Split a path in the optimal solution into two sub-paths



(a) path 1 (b) path 2

Fig. 5. The coverage phase of the path pairs in (a) and (b) are solid line. The contours are oriented horizontally. Their vertical distance to S in the figure corresponds to their distance to S in the environment.

$Retreat$ motion. We call the $Advance$ and $Follow$ motions together as the coverage phase of a path. In the $Follow$ motion, the robot needs two units of energy to visit a cell on the same contour due to the rectilinear motion. For analysis purposes, we do not count the intermediate cells as visited even if it is a newly visited cell. (There is one exception when we count them as covered to get Proposition 5.) The $Retreat$ motion takes up to $\frac{B}{2}$ energy. Then the coverage phase may only visit $(B - \frac{B}{2})/2 = \frac{B}{4}$ cells. But we want the coverage phase of every path in Π_{sol} to visit at least $\frac{B}{2}$ cells. So we let an even-indexed path start from where the previous path retreats and set its coverage phase to start there (line 11 to 13). In our analysis, we pair them and have Proposition 4.

Proposition 4: Let π_{2k+1} and π_{2k+2} be a path pair from Algorithm 1. Their coverage phases together visit at least $\frac{B}{2}$ cells.

Proof: Let π_{2k+1} retreat on the contour C. π_{2k+2}'s coverage phase starts from C and ends at the contour C'. If C' is further than C, as shown in Fig. 5(a), π_{2k+1}'s coverage phase visits $n_1 = d(C) + (\frac{B}{2} - d(C))/2$ cells. π_{2k+2}'s coverage phase visits $n_2 = (d(C') - d(C)) + (\frac{B}{2} - d(C'))/2$ cells. $n_1 + n_2 = \frac{B}{2} + \frac{d(C') - d(C)}{2} > \frac{B}{2}$, which means the coverage phases of π_{2k+1} and π_{2k+2} are able to visit at least $\frac{B}{2}$ cells. When C' is closer than C, as shown in Fig. 5(b), we can get the same result. ■

For simplicity, we use $\Pi_{sol} = \{\pi_1, \pi_2, ..., \pi_m\}$ to denote the output of our algorithm, where each π_i is a path pair from Algorithm 1. The coverage phases of these paths visit at least $\frac{B}{2}$ cells.

The $Advance$ motion of the first path in Π_{sol} is adjacent to the boundary. It enters the contours from one side. Without loss of generality, we call this side the left side. We have the following proposition.

Proposition 5: After every path in Π_{sol}, the covered cells

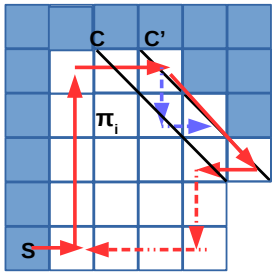


Fig. 6. An example when π_i enters a contour C from the right side, as shown by the red path. The previously covered area is in blue. The blue dashed line denotes its actual trajectory when the path follows the contour C.

on any contour C are all on the left side of C.

Proof: By Algorithm 1, the $Advance$ motion of every path in Π_{sol} enters any contour from the left side. There is one case when the covered cells on a contour are on the right side: A path start to follow a contour from the right side, but it does not have enough energy to finish following this contour, as shown in Fig. 6. The robot uses the $Follow$ motion to finish a contour C' and enters the closer contour C from right side. Then the robot retreats from C to S. When the robot follows the contour C', its actual trajectory switches between C and C', as shown by the dashed line in Fig. 6. So if we count the intermediate cells when following C' as covered, C is fully covered. This is the only case when we count the intermediate cells as covered. Thus the covered cells are all on the left side of any contour. ■

Proposition 5 leads to the following proposition.

Proposition 6: If a path performs the $Follow$ motion on a contour C, any contour C' in $\bar{C}$ (if exists) has uncovered cells no more than C has.

Proof: Let π_i be a path which performs the $Follow$ motion on C, as shown in Fig. 7. According to Proposition 5, the covered cells on C are on the left side of C. Let C_{sub} be the covered subsegment of C. After π_i, let $c_1, c_2, ..., c_p$ be the uncovered cells on C from right to left. c_{p+1} is the rightmost covered cell on C by π_i. Let C' be any contour in $\bar{C}$ and $c'_1, c'_2, ..., c'_q$ be the uncovered cells on C' from right to left. Then c'_{q+1} is the rightmost covered cell on C'. By Proposition 3, we know c'_k is accessible from c_k. If $q > p$, then c'_{q+1} is accessible from a cell which is left to c_{p+1}. Then π_i should stop the $Follow$ motion before reaching c_{p+1} and continue to $Advance$ to the further contours. So $q \leq p$. That is, any contour in $\bar{C}$ (if exists) has uncovered cells no more than C has. ■

With Proposition 6, we can prove the following lemma.

Lemma 1: If a path performs the $Follow$ motion on a contour C, C cannot be a saturated contour.

Proof: Let π_i be a path which performs $Follow$ motion on C. By Proposition 6 we know the contours in $\bar{C}$ has uncovered cells no more than C has. Meanwhile, the following paths' $Advance$ motions will newly cover at least one cell on each contour in $\bar{C}$. So when C is fully covered, the contours in $\bar{C}$ will also be fully covered. So the remaining paths will not visit C anymore. Thus C cannot be saturated.

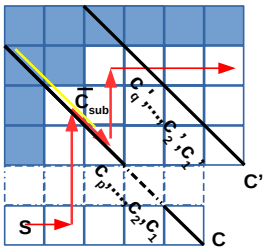


Fig. 7. An example when π_i performs the $Follow$ motion on contour C.

■

Lemma 2: Let C be a saturated contour. For any path in Π_{sol}, when its coverage phase visits C the first time, it has $B - d(C)$ remaining energy to visit at least $\frac{B}{2} - d(C)$ more cells. While for a path in Π^*_{split}, when it reaches C, it can visit at most $\frac{B}{2} - d(C)$ more cells.

Proof: Let π_i be a path in Π_{sol} whose coverage phase visits a saturated contour C. If π_i does not the perform $Follow$ motion before reaching C, it will still have $B-d(C)$ energy when reaching C.

Assume that π_i performs the $Follow$ motion on a contour C' before reaching C. Before π_i, if C and C' has the same number of covered cells (all on left side according to Proposition 5), π_i will not perform the $Follow$ motion on C'. The only possibility is that C has more covered cells than C'. Thus, there should be one path which performs the $Follow$ motion on C. Otherwise, it cannot have more covered cells than C'. According to Lemma 1, C cannot be saturated. It contradicts that C is a saturated contour. The assumption is wrong.

So π_i does not perform the $Follow$ motion before reaching C. The remaining energy is $B-d(C)$ when reaching C. The $Retreat$ motion takes no more than $\frac{B}{2}$ energy. Therefore, when reaching C, π_i can visit $\frac{B}{2} - d(C)$ more cells. Now consider a path in Π^*_{split}. It retreats when its energy drops to $\frac{B}{2}$. So the remaining energy when it reaches C is at most $\frac{B}{2} - d(C)$ and it can visit at most $\frac{B}{2} - d(C)$ cells. ■

Lemma 3: Let $C_1, C_2, ..., C_t$ be the saturated contours and $d(C_1) < d(C_2) < ... < d(C_t)$. Let n_i, n^*_i be the number of paths which reaches $\bar{C}_i$ in Π_{sol} and Π^*_{split}. $n_i \leq n^*_i$.

Proof: Let B_i, B^*_i be the total path length (energy) spent in $\bar{C}_i$ by Π_{sol} and Π^*_{split} respectively. Since C_t is the last saturated contour, $B_t = |\bar{C}_t|$. $|\bar{C}_t|$ is the lower bound of energy needed in $\bar{C}_t$, so $B^*_t \geq |\bar{C}_t|$. Thus $B_t \leq B^*_t$. According to Lemma 2, in $\bar{C}_t$, each path in Π_{sol} newly covers at least $\frac{B}{2} - d(C_t)$ cells while each path in Π^*_{split} covers at most $\frac{B}{2} - d(C_t)$ cells. So $n_t \leq \frac{B_t}{B/2-d(C_t)} \leq \frac{B^*_t}{B/2-d(C_t)} \leq n^*_t$.

Now we prove the Lemma by induction. When $k = t$, $n_t \leq n^*_t$, $B_t \leq B^*_t$. Suppose when $k = j$, we have $n_j \leq n^*_j$ and $B_j \leq B^*_j$. When $k = j - 1$,

$$B_{j-1} = (\bar{C}_j - \bar{C^-_{j-1}}) + B_j + (n_j - |C_j|),$$

$$B^*_{j-1} = (\bar{C}_j - \bar{C^-_{j-1}}) + B^*_j + (n^*_j - |C_j|).$$

Since $B_j \leq B_j^*$, $n_j \leq n_j^*$, we get $B_{j-1} \leq B_{j-1}^*$. By Lemma 2, we know

$$n_{j-1} \leq \frac{B_{j-1}}{B/2 - d(C_{j-1})},$$

$$n_{j-1}^* \geq \frac{B_{j-1}^*}{B/2 - d(C_{j-1})}.$$

Since $B_{j-1} \leq B_{j-1}^*$, we get $n_j \leq n_j^*$. So the statement is also true for k=j-1. Then we can conclude that Lemma 3 is true for all the saturated contours. ■

Theorem 1: Let m be the number of paths from Algorithm 1 and n^* be the number of paths in OPT. Then $m \leq 4n^*$.

Proof: From Lemma 3, it is easy to see that $|\Pi_{sol}| \leq 2n^*$. By Proposition 4, one path in Π_{sol} actually consists of two paths directly from Algorithm 1. So $m \leq 4n^*$. ■

Theorem 2: Let l be the total length of paths from Algorithm 1, and l^* be the total length of OPT. Then $l \leq 8l^*$.

Proof: There are m paths from Algorithm 1. So $l \leq mB$. OPT has n^* paths. No two paths can have a total length less than B. Otherwise they can be combined to form a shorter path. So $l^* \geq \frac{B}{2}n^*$. Since $m \leq 4n^*$, $l \leq 8l^*$. ■

VI. Coverage Strategy for General Environments

The main idea for covering non-contour-connected environments is to partition them into contour-connected parts and run Algorithm 1 on each of them. We first discuss where Lemma 1 fails for non-contour-connected environments. Then we show how to partition the environments and prove that the performance can be bounded by the number of reflex vertices of the environments.

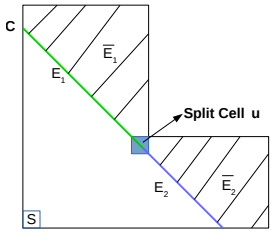
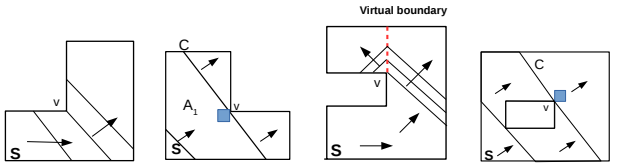


Fig. 8. A non-contour-connected area. The green and blue equi-distance contours are the entrances to the two forked branches from the split cell.

Fig. 8 shows a simple non-contour-connected area. From u, the contour C breaks into two segments E_1 and E_2. $\bar{C}$ forks into two branches, $\bar{E}_1$ and $\bar{E}_2$. Consider the case when $\bar{E}_1$ is small and needs less than $|E_1|$ paths, but $\bar{E}_2$ is large and needs more than $|E_2|$ paths. In this case, the paths will perform the $Follow$ motion before entering $\bar{E}_2$, but the contours in $\bar{E}_2$ can still be saturated. This violates Lemma 1.

Now we explain how to partition the environment into contour-connected subareas. Assume that S is at an acute vertex (If not, we just split the environment into at most four parts by the horizontal and vertical lines which cross S). We use a line of the same direction as the first contour to sweep the environment from S. When the line sweeps to



(a) v is at an endpoint of C (b) v splits C (c) The contours change directions from v (d) Two sweeping processes meet

Fig. 9. Four cases when meeting a reflex vertex during sweeping.

a reflex vertex v on the contour C, there are four possible cases, as shown in Fig. 9. (i) C is a straight line, and v is at an endpoint of C, as shown in Fig. 9(a). In this case, a reflex vertex does not affect the sweeping process. (ii) The contour C at v is a straight line and from v, C breaks into two segments, as shown in Fig. 9(b). We form one contour-connected subarea by the previously swept area. Then we start two new sweeping processes from the two segments of C split by v. In this case, the reflex vertex forms one subarea and starts two new sweeping processes. (iii) The next contours consist of two line segments, as shown in Fig. 9(c). We call the position where the two line segments meet as $turning$ cells. We use these turning cells as a virtual boundary of the environment. We continue the sweeping process at one side of this virtual boundary, and start a new sweeping process from this reflex vertex on the other side of this virtual boundary. In this case, the reflex vertex starts one more sweeping process. (iv) Two sweeping processes meet at v (which implies that v is a vertex of an inner obstacle), as shown in Fig. 9(d). We stop the two sweeping processes to form two subareas. Then we start one new sweeping process from C. In this case, the reflex vertex stops two sweeping processes and starts a new one. When there are no more contours in the current sweeping process, the area swept by this process forms a new subarea. Fig. 10 shows a partition example.

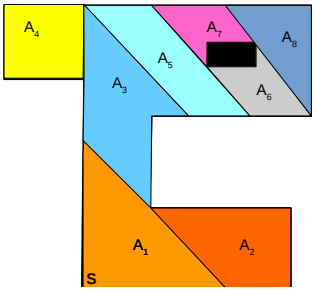


Fig. 10. Division of the environment into contour-connected parts. The reflex vertices which affect the sweeping processes are labeled out as $u1, u2, u3$, and $u4$. The black area represents an inner obstacle.

Let r be the number of reflex vertices of the environment. We run Algorithm 1 on each subarea to get our solution.

Lemma 4: The sweeping processes generate at most $2r+4$ contour-connected subareas of the environment.

Proof: Each reflex vertex starts at most two more sweeping processes. If S is not at an acute corner, there

are at most four sweeping processes from S. So the number of subareas are at most $2r + 4$. ■

Theorem 3: Let n and l be the number of paths and the total path length to cover the given environment by our algorithm. Let n^* and l^* be the number of paths and total path length to cover the same environment by the optimal solution. $n \leq 4(2r + 4)n^*$, $l \leq 8(2r + 4)l^*$.

Proof: Let n_i^* and l_i^* be the number of paths and total path length when OPT covers the $i - th$ subarea from the sweeping process. It is easy to see that $n_i^* \leq n^*$, $l_i^* \leq l^*$. Let n_i and l_i be the number of paths and total path length when Algorithm 1 covers this subarea. According to Theorem 2, $n_i \leq 4n_i^*$, $l_i \leq 8l_i^*$. So $n_i \leq 4n^*$, $l_i \leq 8l^*$. There are at most $2r + 4$ subareas from the sweeping process. Thus

$$n = \sum_{i=1}^{2r+4} n_i = \sum_{i=1}^{2r+4} 4n_i^* \leq \sum_{i=1}^{2r+4} 4n^* = 4(2r + 4)n^*,$$

$$l = \sum_{i=1}^{2r+4} l_i = \sum_{i=1}^{2r+4} 8l_i^* \leq \sum_{i=1}^{2r+4} 8l^* = 8(2r + 4)l^*.$$

■

VII. Field Experiment

The goal of the experiment is to validate the uniform energy consumption model, and to demonstrate the paths from Algorithm 1 in an environmental monitoring application.

Fig. 11. The Experiment filed.

We use a DJI Phantom 3 SE UAV to execute the paths. One reason to use the UAV is that it can fly in four directions (front, back, left, right) without rotation, which allows us to ignore the energy consumed by rotation. We select an open field in Cedar Creek Ecosystem Science Reserve, MN, USA to do the experiment, which is shown in Fig. 11. Most parts of this area are not easily accessible to humans, which makes it ideal for a UAV to cover it. We use the App *Litchi* to guide the UAV to follow the paths. We also read the battery's remaining energy from *Litchi* and use that to measure the energy consumption.

To validate the uniform energy consumption model, we design the flight plan, as shown in Fig. 12 and let the UAV follow it. We record the remaining energy of the battery along the way. We did two flights with the same battery. The straight lines in Fig. 13 illustrate that the uniform energy consumption model is reasonable.

Next, we plan the paths to cover the area. For safety we want the battery to still have enough energy after following each path. We set the energy budget as 1200 meters for each

Fig. 12. The flight plan to test battery consumption.

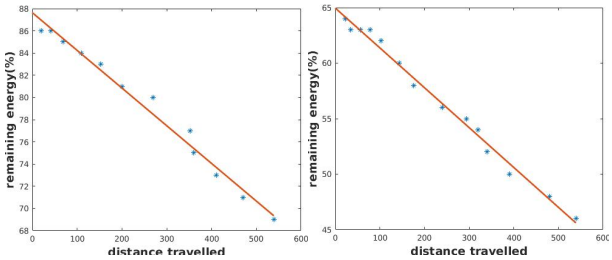


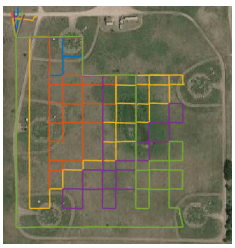
Fig. 13. The battery's remaining energy in terms of distance travelled.

path. The flight altitude is set to be 10 meters, and the UAV's camera is able to cover a $20m \times 20m$ cell. Algorithm 1 returns five paths. The actual flight trajectory is shown in Fig. 14. If we count the cells visited by $Retreat$ motion, the fifth path is unnecessary. The total area is $46400m^2$. One path is able to cover an area of $1200 \times 20 = 24000m^2$ at most. So OPT needs at least 2 paths to completely cover this area. Our solution has five paths, less than four times of OPT, as given by Theorem 1.

VIII. Conclusion and Future Work

In this paper, we presented an algorithm for the energy-constrained coverage path planning problem. For contour-connected environments, the approximation ratio of the algorithm is shown to be 4 for minimizing the path number and 8 for total length. For arbitrary environments, the approximation ratios are $4(2r + 4)$ and $8(2r + 4)$ respectively, where r is the number of reflex vertices of the environment.

(a) path 1 (b) path 2 (c) path 3 (d) path 4 (e) path 5



(f) whole trajectory

Fig. 14. The actual flight trajectory of the UAV. When ploted together in (f), previous paths have higer priority.

In the present work, we restrict the robot motion to be rectilinear. One direction for future research is to extend the algorithm for unconstrained motion. Designing a constant-approximation algorithm for arbitrary problems is another important direction.

ACKNOWLEDGMENT

This work is supported in part by NSF Award # 1525045, a MnDrive RSAM Industrial Partnership grant with The Toro Company, and a grant from Minnesota State LCCMR Program.

REFERENCES

[1] S. Arora. Polynomial time approximation schemes for euclidean tsp and other geometric problems. In *Proceedings 37th Symposium on Foundations of Computer Science*, pages 2–11. IEEE, 1996.

[2] M. Betke, R. L. Rivest, and M. Singh. Piecemeal learning of an unknown environment. *Machine Learning*, 18(2-3):231–254, 1995.

[3] Y. Choi, T. Lee, S. Baek, and S. Oh. Online complete coverage path planning for mobile robots based on linked spiral paths using constrained inverse distance transform. In *International Conference on Intelligent Robots and Systems*, pages 5788–5793. IEEE, 2009.

[4] H. Choset. Coverage of known spaces: The boustrophedon cellular decomposition. *Autonomous Robots*, 9(3):247–253, 2000.

[5] G. N. Frederickson, M. S. Hecht, and C. E Kim. Approximation algorithms for some routing problems. In *Proceedings of 17th Symposium on Foundations of Computer Science*, pages 216–227. IEEE, 1976.

[6] Y. Gabriely and E. Rimon. Spanning-tree based coverage of continuous areas by a mobile robot. In *International Conference on Robotics and Automation*, volume 2, pages 1927–1933. IEEE, 2001.

[7] E. Galceran and M. Carreras. A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258–1276, 2013.

[8] E. Gonzalez, O. Alvarez, Y. Diaz, C. Parra, and C. Bustacara. Bsa: a complete coverage algorithm. In *International Conference on Robotics and Automation*, pages 2040–2044. IEEE, 2005.

[9] G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358, 1992.

[10] Chung-Lun Li, David Simchi-Levi, and Martin Desrochers. On the distance constrained vehicle routing problem. *Operations research*, 40(4):790–799, 1992.

[11] R. Mannadiar and I. Rekleitis. Optimal coverage of a known arbitrary environment. In *International Conference on Robotics and Automation*, pages 5525–5530. IEEE, 2010.

[12] V. Nagarajan and R. Ravi. Approximation algorithms for distance constrained vehicle routing problems. *Networks*, 59(2):209–214, 2012.

[13] I. Shnaps and E. Rimon. Online coverage of planar environments by a battery powered autonomous mobile robot. *IEEE Transactions on Automation Science and Engineering*, 13(2):425–436, April 2016.

[14] H. H. Viet, V. Dang, S. Choi, and T. C. Chung. Bob: an online coverage approach for multi-robot systems. *Applied Intelligence*, 42(2):157–173, 2015.

APPENDIX

Proposition 1:If C is an equi-distance contour in a contour-connected environment, then C is a line segment.

Proof: We prove this proposition by contradiction. Suppose C is an equi-distance contour and is a polyline consisting of multiple segments. Let c be a turning vertex on C, as shown in Fig 15(a). (A turning cell is a cell where two segments on a contour meet, as defined in section VI.) If c is adjacent to the boundary, then c is a split cell by definition. Then the environment is not contour-connected. Let C_{next} be the contour that is one unit further to S than C. If c is not adjacent to the boundary, then C_{next} must

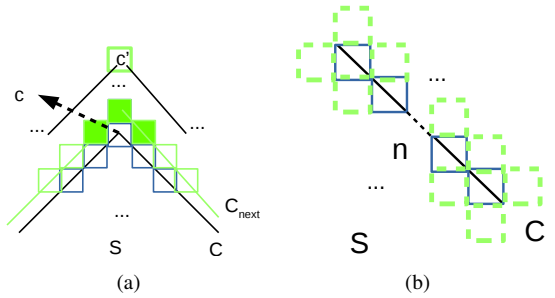


Fig. 15. (a) When the contour C consists of multiple segments, C has turning cells on it. (b) The candidate cell positions for the adjacent contour are the green dashed squares.

have the three cells that are c's neighbors (colored green). So C_{next} is also a polyline consisting of at least two segments and has a turning vertex. Let c' be the furthest turning vertex in $\bar{C}$. Then c' must be adjacent to the boundary. Otherwise, there will be a further turning vertex than c'. So c' is a split cell, and the environment is not contour-connected. Thus, a contour in contour-connected environments must be a line segment. ■

Proposition 2: Let C and C' be two adjacent equi-distance contours. Then $||C| - |C'|| \leq 1$.

Proof: Let C be a contour with n cells. According to Proposition 1, C is a line segment. Let C' be a contour adjacent to C (closer or further to S). The cells C' must be adjacent to cells on C, since a cell on C' must have a unit distance to a cell on C. So there are $(2n+2)$ possible locations for cells on C', as shown in Fig. 15(b). But C' is a line segment, it cannot have $(n+2)$ cells on it. Otherwise, it will have a turning vertex. Neither C' can have less than (n-1) cells, since in that case one cell on C will be adjacent to the boundary and be a split cell, which contradicts the contour-connected assumption. So $||C| - |C'|| \leq 1$. ■

Proposition 3: Let C and C' be two contours in a contour-connected environment and C' is further than C. $c_1, c_2, ..., c_p$ are the cells on C from one side to the other. $c'_1, c'_2, ..., c'_q$ are the cells on C' from the same side to the other. c_i is defined to be c_p if $i > p$. c'_i is defined to be c'_q if $i > q$. Then c'_i is accessible from c_i for all i.

Proof: We first prove that this proposition is true for two adjacent contours. Let C'' be a contour with $d(C'') = d(C) + 1$. Let $c''_1, c''_2, ..., c''_t$ be the cells on C'' starting from the same side as c_1. We prove this by induction.

c''_1 has two candidate positions,c''_{1a} and c''_{1b}, as shown in Fig. 16. Both of them are accessible from c_1. Suppose when $k = j$, c''_j is accessible from c_j. Then there are two possible locations, c''_{ja} and c''_{jb}, for c''_j. They again lead to two possible locations, $c''_{(j+1)a}$ and $c''_{(j+1)b}$, for c''_{j+1}. We can see that both $c''_{(j+1)a}$ and $c''_{(j+1)b}$ are accessible from c_{j+1}. By induction, c''_i is accessible from c_i for all i. The proposition is true for two adjacent contours.

For any two contours C and C' with $d(C') > d(C)$, if all the contours between C and C' have more or equal number of cells, it is easy to see that c'_i is accessible from c_t. Suppose an intermediate contour C_k has k cells, where $k \leq i$, as

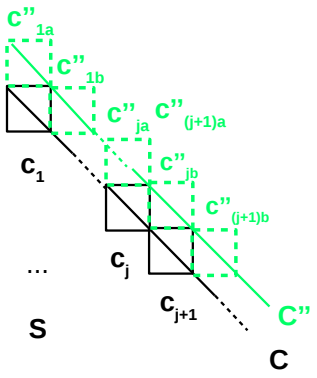


Fig. 16. The position relationship between two adjacent contours.

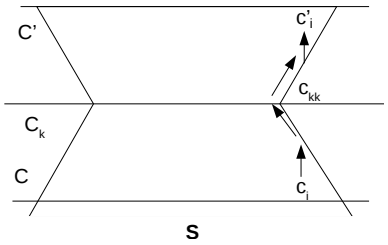


Fig. 17. The case when an intermediate contour has less number of cells.

shown in Fig. 17. We know c_{kk} is accessible from c_i. We can also know that c_i' is accessible from c_{kk}. Thus c_i' is accessible from c_i. The proposition holds for any two contours. ■