2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

Complete and Near-Optimal Path Planning for Simultaneous Sensor-Based Inspection and Footprint Coverage in Robotic Crack Filling

Kaiyan Yu, Chaoke Guo, and Jingang Yi

Abstract— **A simultaneous robotic footprint and sensor coverage planning scheme is proposed to efficiently detect all the unknown targets with range sensors and cover the targets with the robot's footprint in a structured environment. The proposed online Sensor-based Complete Coverage (`online SCC`) planning minimizes the total traveling distance of the robot, guarantees the complete sensor coverage of the whole free space, and achieves near-optimal footprint coverage of all the targets. The planning strategy is applied to a crack-filling robotic prototype to detect and fill all the unknown cracks on ground surfaces. Simulation and experimental results are presented that confirm the efficiency and effectiveness of the proposed online planning algorithm.**

I. Introduction

The simultaneous sensor-based inspection and footprint coverage (SIFC) for autonomous robots is a fundamental problem for many applications. Of particular interest of the SIFC is providing an efficient solution that guarantees the combined purpose of the complete sensor coverage to detect all the unknown targets and the footprint coverage of all the detected targets. The example applications include robotic detecting and filling cracks for civil infrastructure, sensing and cleaning dirty surfaces, and finding and collecting mines.

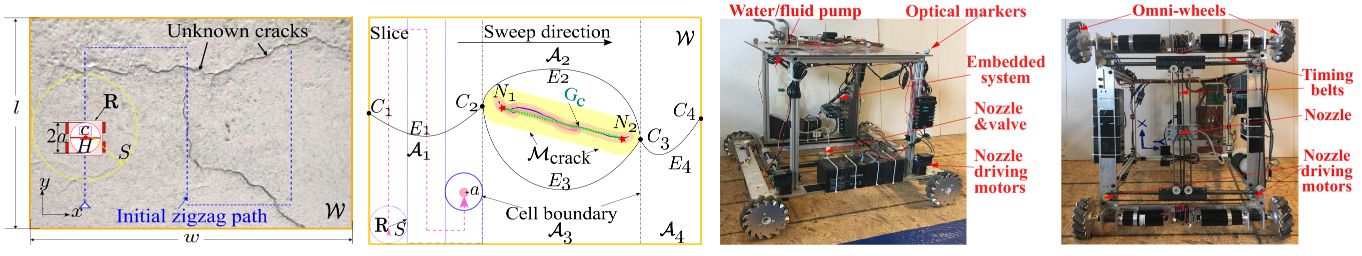
Among these appealing robotic applications, we are particularly interested in the automatic robotic crack filling for civil infrastructure. Detecting and constructing the crack map separately requires extra cost and time before the sealing or filling task because the crack map tends to be different each time. Timely detecting, repairing, and refilling of the cracks is required to prevent their further growth [1]. Efficient and simultaneous detection and recovery of cracks on ground surfaces can significantly reduce the cost and improve safety in road maintenance. In this paper, we demonstrate a complete and near-optimal planning strategy with a crack filling robot to simultaneously detect unknown cracks on the ground and fill them while traveling the shortest distance.

The SIFC planning is related to the *Covering Salesman Problem*, a Traveling Salesman Problem (TSP) variant, in which an agent is required to travel in the shortest distance to visit all specified neighborhood of each city. However, without any knowledge of the locations of each city, the agent must inspect every point in the environment. Other variants include the *Art Gallery Problem*, the complete observability of an art gallery with the minimum number of stationary guards, and the *Watchman Tour Problem*, which computes the shortest route a watchman should take to guard an entire area. However, these problems do not require the footprint coverage of the targets. All of these problems are NP-hard and therefore, obtaining optimal solutions is only feasible for very limited problem domains [2].

To detect unknown targets, robotic exploration can be applied; however, most of the exploration paradigms are not efficient for time-critical applications as the robot may visit the same place more than once during backtracking [3]. Coverage Path Planning (CPP) is widely applied to explore the unknown environment by determining a path that passes over all points of an area of interest while avoiding obstacles [4]. The Morse-based cellular decomposition (MCD) [5] and generalized Boustrophedon decomposition [6] guarantee the complete coverage of an unknown environment. Using omnidirectional range sensor information, the MCD coverage planning methods ensure encountering all the critical points of the decomposition online [7]–[9]. In [10] and [11], the generalized Voronoi Diagram is combined with the MCD to cover narrow and cluttered spaces. A complete coverage of unknown rectilinear environments using a square robot with contact sensing was reported in [12]. Those algorithms performed an on-line decomposition such that the areas could be covered completely by back-and-forth motions. However, there are no claims on the optimality of the planned path. Recent work in [13] proposes an optimal coverage of a known arbitrary environment. The Chinese Postman Problem solution is adapted for the calculation of the coverage sequence by splitting selected cells into two components to eliminate repeat coverage. Grid-based approaches for planning a complete coverage path are also well-studied [14], [15], but those approaches restrict the space and robot motion to the grids. For extensive surveys on the coverage-planning strategies, readers can refer to [4] and [16].

In order to achieve optimal performance, motion planning algorithms have been well studied [17]. However, in practice, it is unrealistic to know detailed environmental information for most situations. Online replanning of the robot path is needed when new information is obtained by the onboard sensors of the robot. A graph-based network simplex method is presented in [18] to solve the replanning problem The method, however, requires the sensing range to be larger than the length of the longest arcs of the graph. The Anytime Dynamic A* algorithm (AD*) [19] is a generic graph-based planning and replanning scheme that can generate bounded suboptimal solutions when the map changes. In general, the start-goal motion planning algorithms do not address the

The work was supported in part by the US NSF under award IIS-1426828.
K. Yu is with the Department of Mechanical Engineering, Binghamton University, Binghamton, NY 13902. C. Guo and J. Yi are with the Department of Mechanical and Aerospace Engineering, Rutgers University, Piscataway, NJ 08854 USA (email: kyu@binghamton.edu; chaoke.guo@rutgers.edu; jgyi@rutgers.edu).



(a) (b) (c) (d)

Fig. 1. (a) The illustration of robotic crack inspection and filling setup with unknown crack information in a structured environment. (b) Illustration of the MCD with a one-edge crack graph. The robot $\mathbf{R}$ has sensing radius of S, and a is its footprint radius. The footprint region is illustrated as the red highlighted area. $\mathcal{A}_i$, C_i and E_i, $i = 1, \cdots, 4$, represent cells, nodes (i.e., critical points) and edges of the Reeb graph, respectively. The target region $\mathcal{M}_{\text{crack}}$ is highlighted in yellow. The crack graph G_c is shown in a dotted green line, and red stars N_1 and N_2 represent the crack graph nodes. (c) The crack filling robot prototype. (d) The bottom view of the crack filling robot.

complete coverage of the space.

Navigation and coverage planning for autonomous underwater vehicles is a closely related problem [20]. However, the existing work only focuses on the unknown environment exploration, and does not usually consider simultaneously detecting and collecting certain targets or performing other, specific tasks while conducting the exploration task. Another closely related problem is the efficient robot cleaning or vacuuming. In [21], the authors proposed Poisson-driven dirt maps to estimate the dynamics of the generation of dirt in a known environment. The learned dirt map is used for planning the robot to clean a set of cells selected for the objective cleaning policies. This problem is different from the problem in this work because the initial learning process is costly or infeasible for applications such as the online construction of the crack map for road rehabilitation.

In our previous work [1], an optimal crack coverage planning was proposed to guide the robot to fill all the cracks with an offline constructed crack graph, which, however, needs all the a priori crack information. To reduce costs and improve efficiency, this paper proposes a complete online strategy, called the online Sensor-based Complete Coverage (`online SCC`) planning algorithm. The `online SCC` can detect unknown cracks using onboard sensors and fill them when the robot traverses the cracks. The algorithm constructs the crack graph and a Reeb graph online to represent the environment. An additional constraint is that the robots footprint must cover all the cracks in the shortest distance.

The contributions of this paper lie in the following aspects: (1) The `online SCC` algorithm guarantees the complete sensor coverage of the free space in the entire environment and the complete footprint coverage of the unknown targets; (2) The total distance traveled by the robot is minimized and the resulting near-optimal path can be calculated in polynomial time; (3) The performance of the planning algorithm is validated and evaluated using a crack filling robot.

II. Problem Formulation

We consider the coverage planning for a robot $\mathbf{R}$ equipped with a range sensor in a compact configuration space $\mathcal{C}$, where $\mathcal{C} \subset \mathbb{R}^2$. The free space of $\mathcal{C}$, denoted as $\mathcal{W}$, is a known area that $\mathbf{R}$ needs to completely cover using its range sensor. In general, $\mathcal{C}$ can be in any shape with a finite number of obstacles. For simplicity, $\mathcal{C}$ is a rectangle area with a size of $l \times w$. The targets in $\mathcal{W}$ are unknown. The *target region*, denoted as $\mathcal{M}$, is defined as the "dilated" area of the targets by the onboard sensor's detecting radius, denoted as S, where $\mathcal{M} \subset \mathcal{W}$. The footprint of $\mathbf{R}$ is defined as the range of the robot effector that needs to cover the targets. $\mathbf{R}$ is assumed to be able to move in any directions in $\mathcal{W}$.

Fig. 1 shows an illustration of the SIFC problem for a crack filling robot. The unknown targets are the cracks and the target region $\mathcal{M}_{\text{crack}}$ is obtained by a dilate crack graph G_c by S. $\mathcal{M}_{\text{crack}}$ illustrates the sensor coverage of the robot traveling along the crack graph G_c. The crack filling mechanism is through the nozzle H mounted on the XY gantry table on $\mathbf{R}$. The nozzle's moving range is the footprint size of the robot, i.e., the πa^2 area underneath the robot, where a is the footprint radius. Further, there is a panoramic camera mounted on the robot to observe the crack information. The panoramic camera has detection radius S.

To focus on the coverage planning problem, the following assumptions are made: (1) Robot $\mathbf{R}$ knows its location in $\mathcal{W}$ using the global positioning system. (2) Crack widths are the uniform in $\mathcal{W}$, and the robot needs the same amount of filling material and time per unit crack length. (3) The cracks in $\mathcal{W}$ is static but unknown. (4) All cracks under the robot's footprint can be filled in time. (5) The sensing range is limited but larger than the robot's footprint, i.e., $S > a$.

We consider that the cost of filling a unit-length of a crack is c_f, and the cost of the robot motion per length is c_m. Our objective is to detect all the cracks in $\mathcal{W}$ and cover the cracks using the footprint with minimized total cost $J = J_f c_f + J_m c_m$, where J_f and J_m are the arc lengths of the robot path when filling and not filling cracks, respectively.

III. Sensor-based Complete Coverage Planning

A. Sensor-based complete coverage with known targets

To better explain our approach, we borrow the following definitions from [5] and [7]. As shown in Fig. 1(b), the *slice* is effectively a vertical line sweeping from left to right along the *sweep direction* in $\mathcal{W}$. A *cell* is an area where slice connectivity does not change, and changes on the connectivity of the slice only occur at *critical points*. A critical point is located on the boundary of an object whose surface normal is perpendicular to the sweep direction. Critical points are used to determine the *cell boundaries*.

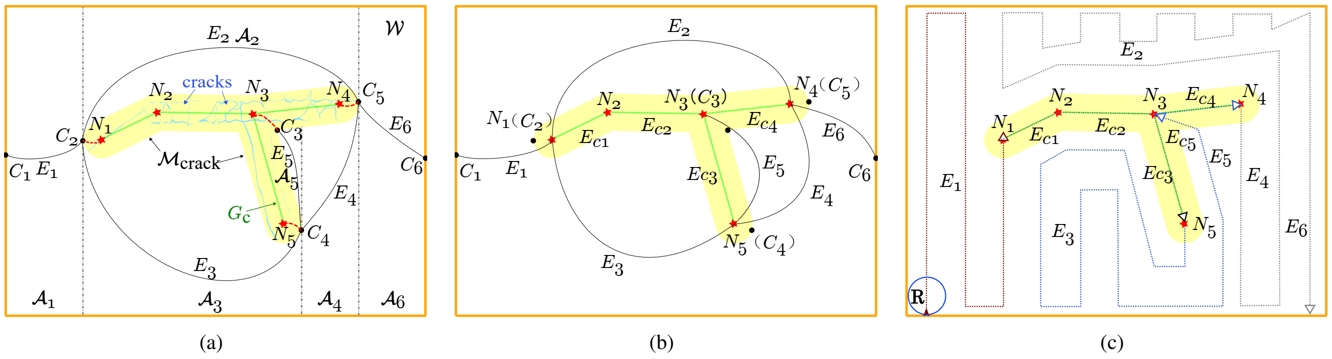


Fig. 2. The illustration of the optimal and complete SIFC planning. (a) The MCD of the configuration space with target region $\mathcal{M}_{\text{crack}}$ (highlighted in yellow). The Reeb graph of the MCD is connected with the crack graph. The red dashed edges are added to the combined Reeb graph and crack graph to form an Euler tour with least cost J. (b) The simplified Reeb and crack graph. Each critical point on the boundary of the target region is combined with its corresponding node in the crack graph. (c) The final path of the robot. The arrows show the traveling direction of $\mathbf{R}$.

Unlike most MCD-based CPP problems, the critical points defined in this paper are not only on the boundary of obstacles in $\mathcal{C}$, but also on the boundary of the target regions $\mathcal{M}_{\text{crack}}$ (i.e., the yellow area in Fig. 1(b)). According to the MCD of free space $\mathcal{W}$, a Reeb graph [5], [7] can be constructed, denoted as G_{w}. In Fig. 1(b), G_{w} is shown as solid lines. The nodes of the Reeb graph represent the critical points C_i and the edges of the Reeb graph, E_i, connect the neighboring critical points. The edges directly correspond to cells $\mathcal{A}_i$, $i = 1, \cdots, 4$.

The crack graph G_c is constructed from the images gathered by the onboard camera. We extract crack skeleton $\mathcal{T}$ from the obtained images and then, $\mathcal{T}$ is dilated by the footprint radius a. The endpoints and intersection points of each crack's dilation are used the build the nodes of G_c, and the crack's extension directions are used to connect edges of G_c. The nodes of the crack graph are at a distance larger than a apart with each other. Multiple endpoints and intersection points are merged if their distances are within a (i.e., those points can be covered by a single footprint of the robot). The edges of G_c are the shortest distance route connecting the corresponding nodes inside the *footprint region* that is defined as the union of all the crack's dilations by the footprint radius. As shown in Fig. 1(b), the footprint region (highlighted in red) describes the union of all the possible robot footprints to cover the whole crack. The nodes and edges of G_c are denoted as N and E_c, respectively. More details on constructing the crack graph can be found in [1].

With the Reeb graph and crack graph, similar to the Chinese Postman Problem, we look for the shortest cost route covering all the graph edges at least once, i.e., we connect edges in both graphs to form an Euler tour with the least cost. In the case of an unspecified ending position, all of the graph nodes except for the initial and ending vertices must have an even number of connected edges (i.e., even degree); otherwise, the robot could get stuck at odd vertices. Therefore, the degree of vertices is made even by adding edges for the robot to repeat or include a part of the area without edge connection. As shown in Fig. 2(a), C_1-C_6, N_1, and N_3-N_5 are the odd vertices, and the red dashed edges connecting $\{C_2,N_1\}$, $\{C_3,N_3\}$, $\{C_5,N_4\}$, and $\{C_4,N_5\}$ are added to the graph to form the shortest route. The resulting Euler tour, denoted as π, provides an order of edges that the robot should visit. Finally, the complete coverage is performed following the sequence of edges in the Euler tour.

The back-and-forth motion path is generated for covering the interior of the cell sequences using the range sensor. By following the crack graph edges, the cracks are covered by the robot footprint. The back-and-forth coverage motion is well documented in the literature [5], [16]. If an edge of the Reeb graph is doubled in the resulting Euler tour, the corresponding cell is split in half, and the robot is capable of controlling the exit point from each cell by varying the height of the coverage progressively [13]. Thus, by minimizing the distance from the exit point to the entry point of the next cell, the connecting distance of each cell can be minimized. Algorithm 1 briefly describes the Sensor based Complete Coverage (SCC) planning. As the Reeb and crack graphs provide a complete model of $\mathcal{W}$, and each edge of the Euler tour is traversed exactly once by definition, the proposed algorithm guarantees optimal and complete coverage of all the cracks and the whole free space $\mathcal{W}$ in terms of minimizing the traveling distance cost J.

Algorithm 1: SCC

```
Input  : $\mathcal{W}, \mathcal{T}, S$
Output: Path
1 $G_{\text{w}} \leftarrow$ Reeb_Graph(MCD($\mathcal{W}$))
2 $G_{\text{c}} \leftarrow$ Crack_Graph($\mathcal{T}$)
3 $\pi \leftarrow$ Euler_Tour($G_{\text{w}}, G_{\text{c}}, J$)
4 Path $\leftarrow$ Complete_Coverage($\pi, S$)
```

B. Online SCC *planning with unknown target information*

When the target information is unknown, the robot needs to detect cracks and critical points online in a given $\mathcal{W}$. Note $\mathcal{W}$ may contain obstacle boundaries, thus the Reeb graph of $\mathcal{W}$ also includes obstacle information. We first combine the crack graph and the Reeb graph using the following lemmas.

Lemma 1: Each critical point generated by the target region $\mathcal{M}_{\text{crack}}$ corresponds to one node of the crack graph.

Lemma 2: Optimality is preserved with the choice of connecting critical points of target region $\mathcal{M}_{\text{crack}}$ to their corresponding nodes in the crack graph.

See the Appendix for the proof of Lemma 1 and 2.

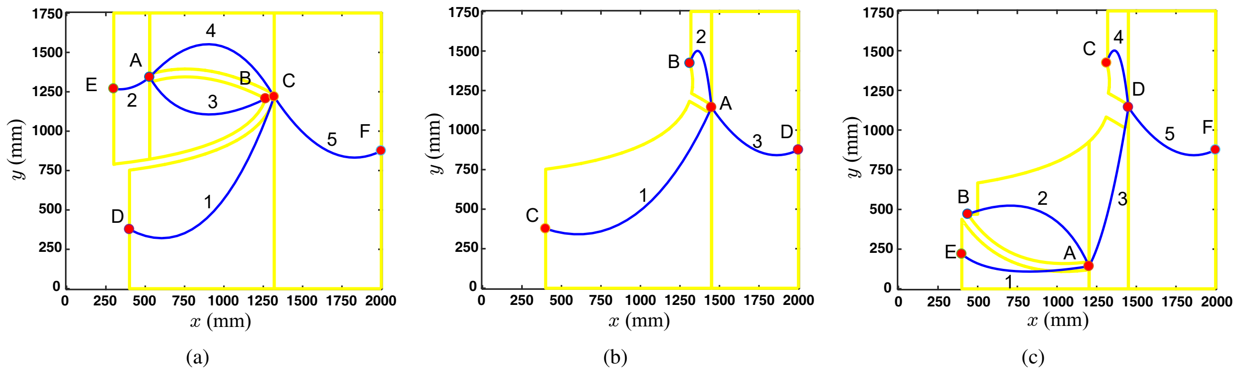


Fig. 3. The Reeb graphs during the loop cycles of the `online SCC` algorithm as more regions are detected. The whole configuration is shown in Fig. 4(a). Each Reeb graph is updated when the robot is on point A. Those Reeb graphs are constructed after removing already covered areas. The letters and red dots denote the node of the graph. The numbers and blue lines are the edges. The highlighted boundaries are the cell boundaries.

According to Lemma 2, we simplify the Reeb graph and crack graph by combining the critical points of the target region to their corresponding nodes of the crack graph. Fig. 2(b) shows the simplified graph of Fig. 2(a). Based on the simplified graph, we propose the `online SCC` planning algorithm without known target information. The robot treats the target region as other obstacles until it finds those combined critical points to enter the target region and follows the crack graph according to the current Euler tour.

Algorithm 2: `online SCC`

```
   Input  : $\mathcal{W}, S$
   Output: $Path$
1  $\mathcal{M}_{\text{cover}} \leftarrow \emptyset$, $\mathcal{T} \leftarrow \emptyset$
2  $Path \leftarrow$ SCC$(\mathcal{W}, \mathcal{T}, S)$
   while $\mathcal{M}_{cover} \neq \mathcal{W}$ do
       if critical point of crack $\mathcal{T}$ is found then
3          $G_{\text{c}} \leftarrow$ Crack_Graph$(\mathcal{T})$
4          FollowPath$(G_{\text{c}})$
5          $\mathcal{M}_{\text{cover}} \leftarrow \mathcal{M}_{\text{cover}} \cup (G_c \oplus S)$
6          $Path \leftarrow$ SCC$(\mathcal{W} \backslash \mathcal{M}_{\text{cover}}, \mathcal{T}, S)$
7      $P_n \leftarrow$the next step of $Path$
8      FollowPath$(P_n)$
9      $\mathcal{M}_{\text{cover}} \leftarrow \mathcal{M}_{\text{cover}} \cup (P_n \oplus S)$
```

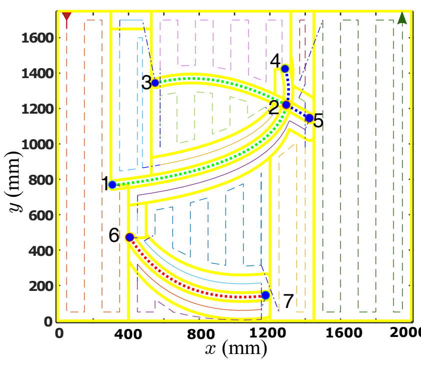
Algorithm 2 shows the structure of the `online SCC` planning algorithm. The robot stores and incrementally constructs the combined graphs online. The already covered area is denoted as $\mathcal{M}_{\text{cover}}$, which is empty at very beginning (line 1). The robot $\mathbf{R}$ follows the initial optimal path calculated by Algorithm 1 in a known environment $\mathcal{W}$ but without any crack information $\mathcal{T}$ until a node of crack graph is encountered (line 2). The robot enters the target region to construct the crack graph (line 3) and follows the crack graph until it ends (line 4). The robot updates the covered area $\mathcal{M}_{\text{cover}}$ (line 5), where operation $\oplus$ represents the dilation calculation, i.e., the Minkowski sum. $\pi \oplus S$ gives the area by dilating path π by the radius of S. Then, Algorithm 1 is used to calculate the optimal path given the current uncovered area $\mathcal{W} \backslash \mathcal{M}_{\text{cover}}$ and detected crack information (line 6). Then, the robot follows the next uncovered edge from the calculated *Path* to cover the corresponding cell (line 7 and 8). If the critical point is connected with more than one uncovered cells, the robot picks the one that has the least number of successive edges (children) in the current Reeb graph. When there are no uncovered cells or edges remaining in the Reeb graph, the environment is optimally and completely covered.

It is worth noticing that to avoid passing the crack twice to reach another uncovered cell, in Algorithm 2, after traversing one crack edge, we remove the covered areas (see Fig. 3 as examples) and update the Reeb graph of the remaining space $\mathcal{W} \backslash \mathcal{M}_{\text{cover}}$. When the resulting Euler tour needs to double edges in the Reeb graph, the corresponding cell is split into two components. The first part is covered by a wall-following motion (the wall is defined as the boundaries of the $\mathcal{W} \backslash \mathcal{M}_{\text{cover}}$). The other part is covered by the zigzag motion of the leftover space in the cell. As shown in Fig. 3(a), the robot is currently on node A (current starting point), and therefore, node A should be odd. Edges 2, 3, and 1 are doubled to form the least cost Euler tour. Similarly, edge 1 is doubled in Fig. 3(b) and edges 1, 2, and 4 are doubled in Fig. 3(c). We select the coverage motion direction according to the next connected edge in the path to minimize the cost.

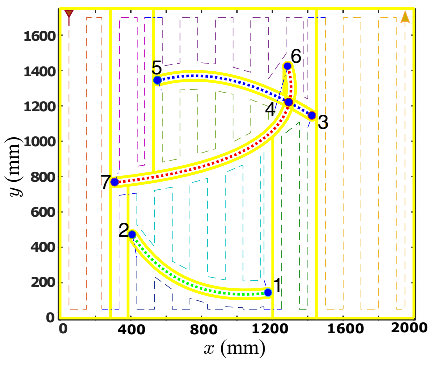
Lemma 3: The `online SCC` algorithm guarantees the completeness of coverage planning and its optimality in terms of the cell visiting sequence.

Sketch of Proof: The completeness of the `online SCC` algorithm follows directly from the properties of the Euler tour used to solve for the route. By definition, the construction of the Reeb graph does not stop until all the areas in $\mathcal{W}$ is covered. The Reeb graph provides a complete representation of the configuration space. Because each edge of the graph is traversed (i.e., each cell is covered), we guarantee that all available free space has been covered. Therefore, the proposed algorithm is complete.

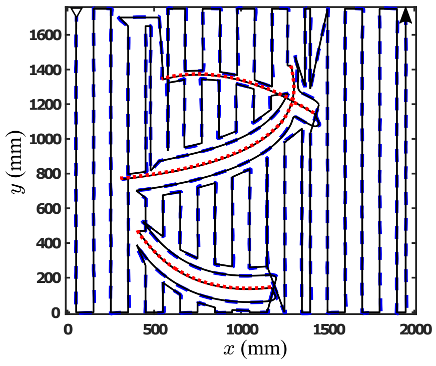
The `online SCC` algorithm reinforces the connections of the critical points to their corresponding nodes in the crack graph by following the crack graph until it reaches an end. Notice that the covered spaces are removed from $\mathcal{W}$, and the cell coverage does not duplicate any covered area. Doubling the edges of cells does not increase any cost because the two components of the cells do not overlap. By definition, no edge of the Euler tour is traversed twice, and this implies no area is covered twice. Therefore, the proposed algorithm



(a)



(b)



(c)

Fig. 4. (a) Simulation result using the `online SCC` algorithm. The crack graph shown in dotted lines are unknown in $\mathcal{W}$ initially. The final MCD of the configuration space is plotted in highlighted yellow. The final trajectory is shown in dashed lines and the dotted dash lines are the connections between two cells. The robot starts from the top-left corner and ends at the top-right corner, marked by triangle and arrow, respectively. When there is a doubled edge, the corresponding cell is covered by a wall-following path and a zigzag path as shown in solid line and dashed line, respectively. (b) The final route of the robot with known crack information. (c) Experimental result using the `online SCC` algorithm. The dashed lines are the robot trajectories, and the overlaid solid lines are the online planned path.

is optimal in the sense of visiting sequence, as all free space is covered exactly once. ■

Because the dimensions of the cells are unknown, the robot does not always have enough information to minimize the zigzag motion connecting adjacent cells according to the Euler tour sequence. As a result, the final trajectory is near optimal. The difference from the optimal solution is bounded by the slice length times the number of the free cells in the Reeb graph. The simplified Reeb graph reduces the number of edges. Both of the algorithms are solved in polynomial time because of the structure of the Reeb graph, therefore, they can be used online efficiently.

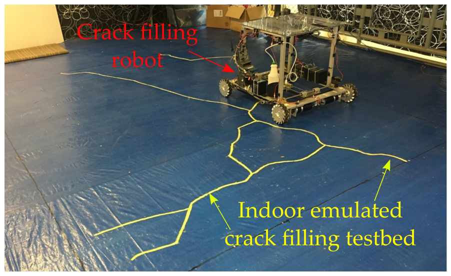


Fig. 5. The experimental setup with crack filling robot prototype on the indoor emulated crack filling testbed.

IV. Simulation and Experimental Results

A. Simulation results

As shown in Fig. 4, we consider a rectangular free space $\mathcal{W}$ with $l = 1750$ mm and $w = 2000$ mm. The radii $S = 50$ mm and $a = 20$ mm. The crack information (dotted lines) is unknown initially to the robot. The robot first follows the initial path planned by Algorithm 1 until the robot detects a crack node at node 1. Then it follows the crack graph to the intersection node 2, and randomly chooses any one of the edges at the intersection. Here the robot chooses the green dotted path and follows it to reach node 3. The covered regions are removed from $\mathcal{W}$, and the Reeb graph and crack graph are updated accordingly. Fig. 3 shows the updated graphs. For all three instances, the robot is on point A, and it chooses the edge with the least number of children in the Reeb graph to traverse next. Fig. 4(a) shows the final path.

In the same configuration, the SCC algorithm is used to provide the ground truth planning result with known crack information. The least cost Euler tour is constructed by connecting node 6 and the concave node near 4 in Fig. 4(b). The final route of the robot is shown in Fig. 4(b).

Table I demonstrates performance comparison of the ground truth planning result with known target information using the SCC algorithm and the online planning result using the `online SCC` algorithm with unknown target information. All the algorithms in the table completely cover $\mathcal{W}$ using sensor. The "Sensor exploration" method generates zigzag motion to solve the complete sensor coverage problem. It results in least sensor overlap area in the whole space, but cannot guarantee the complete crack coverage. The "Footprint coverage" presents the exhaustive coverage with robot footprint, but it generates the most overlaps. The comparison shows the proposed `online SCC` algorithm completely covers the entire free space using sensor and fills all the cracks, and the total traveling distance and traveling time are very close to the grand truth, i.e., the "SCC". Note if the connections between cells are ignored, the "`online SCC` w/o con." performs almost the same as the ground truth "SCC w/o con". Both proposed algorithms outperform the exhaustive footprint coverage and the sensor exploration.

TABLE I

Performance comparison for the configuration in Figure 4

	Crack coverage	Overlap	Distance (m)	Time (s)
Sensor exploration	36.5%	0%	34.9	392.0
Footprint coverage	100%	44.5%	126.4	1395.8
`online SCC`	100%	18.4%	41.5	535.6
SCC	100%	14.4%	40.1	517.3
`online SCC` w/o con.	100%	10.5%	38.7	487.7
SCC w/o con.	100%	11.0%	38.9	492.6
Experiment	100%	23.0%	48.1	595.5

B. Experimental result

Fig. 5 shows the experimental setup with crack filling robot on the indoor emulated crack filling testbed. During the experiment, the maximum robot linear speed is set to be 100 mm/s, and the maximum linear acceleration is 100 mm/s^2.

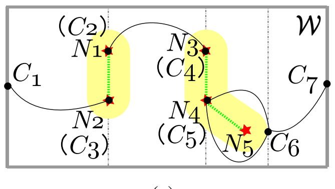
The online SCC algorithm plans and updates the path in real time. The planned path provides the position and speed trajectory of the robot considering the maximum speed and acceleration constraints. The trajectory is transferred to the low-level motion controller through User Datagram Protocol. The kinematic model of the robot is used to conduct the motion control of the robot. The robot positioning information is provided by a motion capture system. Fig. 4(c) shows the experimental result using the online SCC algorithm with unknown crack information. Table I shows the performance comparison of the online SCC experimental result with the simulation results. The experimental result confirmed the efficiency and effectiveness of the proposed online SCC algorithm to conduct online planning for the SIFC tasks.

V. Conclusion

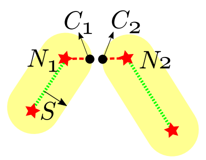
In this paper, we presented a novel algorithm for SIFC problem with unknown target information. The algorithm guaranteed the complete coverage of the structured environment with near-optimal performance in traveling distance. The planning strategy constructed the unknown target graph and Reed graph online, and the solution can be calculated in polynomial time. Simulation and experimental results confirmed the effectiveness of the proposed algorithm. The presented near-optimal and complete coverage planning algorithm can be potentially applied to many other applications in civil infrastructure and public services.

Appendix I
Sketch Proof of Lemma 1

From the assumption of the MCD, no two critical points change the slice connectivity at the same time. Thus, critical points that are collinear with the slice direction need special consideration. Let an endpoint node be a node that has only one connected edge. With horizontal sweep direction, we treat vertical crack edges connecting with endpoint nodes as cell boundaries and those endpoint nodes as critical points, denoted as vertical critical points. As shown in Fig. 6(a), N_1, N_2, and N_3 are vertical critical points. Since the vertical edge of the crack graph separates one cell into two adjacent cells, vertical critical points have two edges in the Reeb graph. One edge comes from the crack graph and the other one comes from one of the two adjacent cells. With this extension, the slice connectivity remains constant within each cell.



(a)



(b)

Fig. 6. (a) Special consideration is given to the vertical critical points, i.e., the endpoint nodes N_1, N_2, and N_3. (b) If the distance between two critical points C_1 and C_2 is less than S, then the distance between the corresponding nodes N_1 and N_2 in the crack graph is greater than $2S$.

Except for the vertical critical points, the remaining critical points of the target region are generated by the surface normal that is perpendicular to the sweep direction. Note the target region is the "dilated" crack graph by S. If the boundary of the target region is convex, then we can always find a node in the crack graph within S distance of the critical point. As shown in Fig. 2(a), critical points C_2, C_4, and C_5 correspond to nodes N_1, N_5, and N_4 in the crack graph, respectively. If the boundary of the target region is concave, then the corresponding node in the crack graph is the intersection point of edges (intersection node), e.g., critical point C_3 is associated to node N_3 in Fig. 2(a). Therefore, every critical point on the boundary of the target region is associated with one node in the crack graph.

Appendix II
Sketch Proof of Lemma 2

Because of the nature of the MCD, all the critical points that are generated by the convex boundaries (i.e., convex critical points) are connected to three cells. Similarly, all the critical points that are generated by the concave boundaries (i.e., concave critical points) are connected to one cell. In the Reeb graph, the convex and concave critical points correspond to nodes of degree three and degree one, respectively. Let us denote the convex and concave critical point of the target region as C_{vex} and C_{cav} and their corresponding crack node according to Lemma 1 as N_{vex} and N_{cav}, respectively.

If N_{vex} is an endpoint node, then it has degree one. If N_{vex} is an intersection node, then its degree plus the number of concave critical points associated with N_{vex} is odd. In order to form the Euler tour, the nodes with odd degree need to be paired up with least cost. Notice that the distance between C_{vex} and N_{vex} is the sensing range S. Because the distance of other nodes in the crack graph to C_{vex} is greater or equal than S, connecting each pair of C_{vex} to N_{vex} result in parts of the minimum cost Euler tour. For the case that the distance between two critical points C_1 and C_2 is less than S, as shown in Fig. 6(b), the distance between the corresponding nodes N_1 and N_2 in the crack graph is greater than $2S$. Therefore, it is optimal to combine the C_{vex} (node of degree three) to their corresponding N_{vex} (odd degree node) in the Euler tour. Those combined nodes have even degree guaranteeing that the robot is not stuck at such nodes.

For C_{cav}, N_{cav} must be an intersection node. Notice that the parity of the intersection node of the crack graph is the same as the parity of the number of critical points associated with it. Therefore, pairing up these odd nodes among them with least cost result in the optimal Euler tour.

For the vertical critical points, they are defined on the graph nodes in Lemma 1 and have degree two. The vertical critical points do not affect the optimality of the Euler tour.

Therefore, to find the minimum cost Euler tour, the edge of the crack graph never gets doubled as all the critical points of target region result in even degrees by connecting them to their corresponding nodes in the crack graph. To pair up other odd nodes in the Reeb graph, only edges corresponding to the cells (free space) get doubled. Doubling of the selected edges means splitting corresponding cells into two portions, which does not increase the cost to cover the whole area. Thus, as all parts of the free space and crack graph are covered exactly once, the optimality is preserved.

REFERENCES

[1] C. Guo, K. Yu, and J. Yi, "Optimal motion planning and control of a crack filling robot for civil infrastructure automation," in *Proc. IEEE Conf. Automat. Sci. Eng.*, Xian, China, 2017, pp. 1463–1468.

[2] R. Bormann, F. Jordan, J. Hampp, and M. Hägele, "Indoor coverage path planning: Survey, implementation, analysis," in *Proc. IEEE Int. Conf. Robot. Autom.*, Brisbane, Australia, 2018, in press.

[3] A. T. Palacios, A. Sánchez L, and J. M. E. Bedolla Cordero, "The random exploration graph for optimal exploration of unknown environments," *Int. J. Adv. Robot. Syst.*, vol. 14, no. 1, p. 1729881416687110, 2017.

[4] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Rob. Auton. Syst.*, vol. 61, no. 12, pp. 1258–1276, 2013.

[5] E. U. Acar, H. Choset, A. A. Rizzi, P. N. Atkar, and D. Hull, "Morse decompositions for coverage tasks," *Int. J. Robot. Res.*, vol. 21, no. 4, pp. 331–344, 2002.

[6] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *Proc. Int. Conf. on Field and Service Robotics*, Canberra, Australia, 1998, pp. 203–209.

[7] E. U. Acar and H. Choset, "Sensor-based coverage of unknown environments: Incremental construction of morse decompositions," *Int. J. Robot. Res.*, vol. 21, no. 4, pp. 345–366, 2002.

[8] E. U. Acar, H. Choset, and J. Y. Lee, "Sensor-based coverage with extended range detectors," *IEEE Trans. Robotics*, vol. 22, no. 1, pp. 189–198, 2006.

[9] E. Acar, H. Choset, Y. Zhang, and M. Schervish, "Path planning for robotic demining: robust sensor-based coverage of unstructured environments and probabilistic methods," *Int. J. Robot. Res.*, vol. 22, no. 7-8, pp. 441–466, 2003.

[10] H. Choset, S. Walker, K. Eiamsa-Ard, and J. Burdick, "Sensor-based exploration: Incremental construction of the hierarchical generalized voronoi graph," *Int. J. Robot. Res.*, vol. 19, no. 2, pp. 126–148, 2000.

[11] H. Choset and J. Burdick, "Sensor-based exploration: The hierarchical generalized voronoi graph," *Int. J. Robot. Res.*, vol. 19, no. 2, pp. 96–125, 2000.

[12] Z. J. Butler, A. A. Rizzi, and R. L. Hollis, "Contact sensor-based coverage of rectilinear environments," in *"Proc. 1999 IEEE Int. Symp. Intelligent Control Intelligent Systems and Semiotics*, 1999, pp. 266–271.

[13] R. Mannadiar and I. Rekleitis, "Optimal coverage of a known arbitrary environment," in *Proc. IEEE Int. Conf. Robot. Autom.*, Anchorage, AK, USA, 2010, pp. 5525–5530.

[14] A. Zelinsky, R. A. Jarvis, J. Byrne, and S. Yuta, "Planning paths of complete coverage of an unstructured environment by a mobile robot," in *Proc. Int. Conf. Adv. Robot.*, vol. 13, 1993, pp. 533–538.

[15] Y. Gabriely and E. Rimon, "Spiral-stc: An on-line coverage algorithm of grid environments by a mobile robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 1. IEEE, 2002, pp. 954–960.

[16] H. Choset, "Coverage for robotics–a survey of recent results," *Ann. Math. Artif. Intell.*, vol. 31, no. 1-4, pp. 113–126, 2001.

[17] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006.

[18] T. Ersson and X. Hu, "Path planning and navigation of mobile robots in unknown environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, vol. 2, 2001, pp. 858–864.

[19] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic A*: An anytime, replanning algorithm." in *Proc. Int. Conf. Automated Planning and Scheduling*, 2005, pp. 262–271.

[20] L. Paull, S. Saeedi, M. Seto, and H. Li, "Sensor-driven online coverage planning for autonomous underwater vehicles," *IEEE/ASME Trans. Mechatronics*, vol. 18, no. 6, pp. 1827–1838, 2013.

[21] J. Hess, M. Beinhofer, D. Kuhner, P. Ruchti, and W. Burgard, "Poisson-driven dirt maps for efficient robot cleaning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 2245–2250.