2016 IEEE International Symposium on
Safety, Security, and Rescue Robotics (SSRR)
EPFL, Lausanne, Switzerland, October 23-27, 2016

Coverage Path Planning for Legged Robots in Unknown Environments

Dan Jia[1], Martin Wermelinger[1], Remo Diethelm[1], Philipp Krüsi[2], Marco Hutter[1]

Abstract— **In applications like search and rescue, industrial inspection, or construction site mapping, robots may be used to explore and map unknown areas using onboard sensing. This paper presents a novel coverage path planning (CPP) algorithm for autonomous exploration with legged robots. The algorithm decomposes the region of interest into cells by detecting landmarks in the environment. Each cell is covered using a zig-zag motion pattern. The landmark detection is based on the traversability information of the surrounding area. The detection mechanism can be applied to obstacles with any shape, and it is robust against measurement uncertainty. Unlike existing CPP algorithms, the proposed algorithm takes into account the traversability of the surrounding terrain, and it is applicable to rough terrain environments where obstacle detection is a non-trivial problem. Completeness of exploration and the trade off between completeness and efficiency are discussed. Two simulations of the proposed algorithm were conducted using the open source robotics simulator *Gazebo*, with the legged robot *ANYmal* [1] as robotic platform. The results demonstrate the feasibility of the proposed algorithm for exploring unknown, rough terrain environments.**

I. Introduction

Legged robots are well suited for navigation in rough and unstructured terrain. Compared to their size, they are generally able to overcome larger obstacles than wheeled or tracked vehicles. Moreover, legged robots are typically holonomic and thus omnidirectional, which facilitates navigation in intricate environments, such as places cluttered with obstacles. Equipped with adequate sensing instruments, legged robots can be used for exploring initially unknown areas in applications such as search and rescue, industrial inspection, construction site mapping, etc.. For many of these applications, completeness of the exploration is desired. Thus, a coverage path planning (CPP) algorithm is required. Ideally, the algorithm should yield complete coverage of the prescribed region with minimal expenses in terms of time, energy, or traveled distance.

Most existing CPP approaches assume that the robot has a sensor (or effector) range roughly equal to the size of the robot, and that covering a region requires the robot to reach every point within the region. This is a valid assumption for applications like vacuum cleaning, demining, or painting. However, for an exploration task, the robot is equipped with a sensor whose range extends well beyond the size of the robot. Coverage can be achieved without physically visiting

This work was supported in part by the Swiss National Science Foundation (SNF) through project 200021_149427 / 1, the National Centre of Competence in Research Robotics, and the armasuisse S+T UGV research program.

[1]Robotic Systems Lab (RSL), ETH Zurich, Switzerland, jiad@student.ethz.ch

[2]Autonomous Systems Lab (ASL), ETH Zurich, Switzerland

Fig. 1. The legged robot *ANYmal*, which was used as robotic platform for simulations.

every point in the environment, as long as the entire region has been observed by the sensor.

In this paper we present a novel online coverage path planning algorithm for exploring initially unknown areas. The algorithm decomposes the region to be covered into cells, and covers each cell using a zig-zag pattern. The decomposition is based on landmarks in the environment, which are detected using the estimated terrain traversability. The latter is computed with a method specifically designed for legged robots, which was presented in our previous work [2]. The proposed CPP algorithm can be applied to robots equipped with long-range sensors. Moreover, it takes into account information about the traversability of the surrounding terrain, and it does not make any assumptions about the geometry of obstacles. As a result, the algorithm can be used in rough terrain environments where obstacle outlines cannot be readily extracted from range sensor measurement. Two simulated experiments with our legged robot *ANYmal* demonstrate the feasibility of the CPP algorithm for exploration of unknown and rough terrain environments.

The remainder of this paper is structured as follows. Section II presents a review of previous work on coverage path planning. Section III describes our coverage path planning algorithm. Section IV presents the results of simulated experiments. Finally, we summarize and conclude in Section V.

II. Related Work

To accomplish complete coverage, previous works in CPP decompose the region of interest into cells, either explicitly or implicitly. Depending on the decomposition method used, Choset [3], [4] classified these algorithms into approximate cellular decomposition, semi-approximate decomposition, and exact cellular decomposition.

Approximate cellular decomposition discretizes the region to be covered into a grid map, whose cell size is usually chosen to be equal to the size of the robot's footprint or sensor range. The obstacle occupancy of neighborhood cells is evaluated when the robot reaches a new cell, and unoccupied neighborhood cells are stored. Complete coverage is achieved by visiting all unoccupied cells in the grid map. Two examples using approximate cellular decomposition are the *Wavefront* algorithm [5] and *Spiral-STC* (Spanning Tree Coverage) [6], [7], with the former being an offline algorithm, and the latter being an online algorithm that can be applied in unknown environments. By construction, these algorithms are ideal only for applications with short sensor range, which results in grid cells small enough to represent the environment accurately. An exception is the complete *coverage D** algorithm developed by Dakulović *et al.* [8], which allows the robot to occupy multiple grid cells at the same time. However, the algorithm involves graph search and can be computationally demanding for large grid maps.

Semi-approximate decomposition also discretizes the region into cells, but the discretization only happens in one direction. In other words, for a semi-approximate decomposition, the cell has fixed width and arbitrary top and bottom. Hert and Lumelsky [9] presented an algorithm for autonomous underwater terrain coverage in three-dimensional unknown space using semi-approximate. Same as for approximate cellular decomposition, due to the discretization involved in one direction, these approaches are not ideal when the robot sensor is omnidirectional and has a range that extends beyond the robot size.

Exact cellular decomposition, on the other hand, decomposes the region into cells whose geometry exactly matches the environment without discretization. The robot covers each cell using a simple motion pattern. The decomposition is based on landmarks in the environment. Two early offline approaches, *trapezoidal decomposition* [10] and *boustrophedon decomposition* [11], use vertices of polygons as landmarks. As a result, they can only be applied to polygonal obstacles. Generalizing the boustrophedon decomposition, Acar *et al.* [12], [13], [14] developed *Morse decomposition*, which uses critical points of Morse functions as environment landmarks. Morse-based decomposition fails when the critical points are ill-defined, i.e., when the gradient of the obstacle outline does not exist, or the obstacle outline is parallel to the Morse function. Wong and MacDonald [15], [16] developed *slice decomposition*. It uses an imaginary sweeping slice to sweep through the region. When an obstacle is encountered, the sweeping slice is split into three segments, with two segments in free space, and one segment inside the obstacle. The changing of segment numbers is used as environment landmark to decompose regions into cells. Both Morse-based decomposition and slice decomposition are designed for robots with short sensor range. Addressing this issue, Acar *et al.* [17], [18] presented a coverage algorithm that can be applied on robots with long sensor range by hierarchically combining Morse-based decomposition and Voronoi diagrams. However, no experimental verification of the proposed approach is found in the literature.

III. Exploration Coverage Path Planning Algorithm

Our CPP approach uses exact cellular decomposition for planning on the global level, and approximate cellular decomposition for landmark detection in a local area within the sensing range. We represent the global cells by *nodes*, which are defined as coordinate points within the region to be covered. When a landmark is detected, a node is added to the *node set*. On the global level, the robot covers the region using a simple zig-zag pattern (known as boustrophedon motion), and travels to a node in the node set when the current zig-zag pattern is finished. The sweeping distance for the zig-zag pattern is smaller or equal to the effective sensor range. On the local level, the area within the current range is decomposed into grid cells. Each grid cell is evaluated for obstacle occupancy, and the results are used to detect landmarks.

In our previous work, we developed a terrain traversability estimation method for legged robots [2], which generates a traversability grid map of the robot's surroundings. The traversability grid map has cell values $t \in [0, 1]$, with $t = 0$ representing a non-traversable cell, and $t = 1$ representing a perfectly traversable cell. The proposed CPP algorithm uses this traversability grid map for landmark detection. There are two advantages of using such a traversabiliy map. First, it allows us to incorporate the actual traversability of the terrain for a specific robot. In contrast, existing CPP approaches typically detect landmarks solely based on range or contact sensor measurements, which do not necessarily reflect terrain traversability for complex systems like legged robots. Second, utilizing omnidirectional traversability information, our approach enables starting the exploration from any point within the region. In contrast, existing approaches such as [15] or [16] require the robot to reach a region corner before starting the coverage, which can be difficult in complex environments.

A. Global Path Planning

The path planning on the global level is realized as a finite state machine with two states: *normal* and *travel*. In *normal* state, the robot moves on a zig-zag pattern, and detects landmarks using the traversability information of the

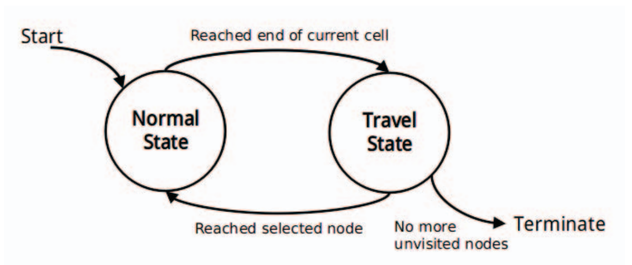


Fig. 2. Two state iteration of the main algorithm.

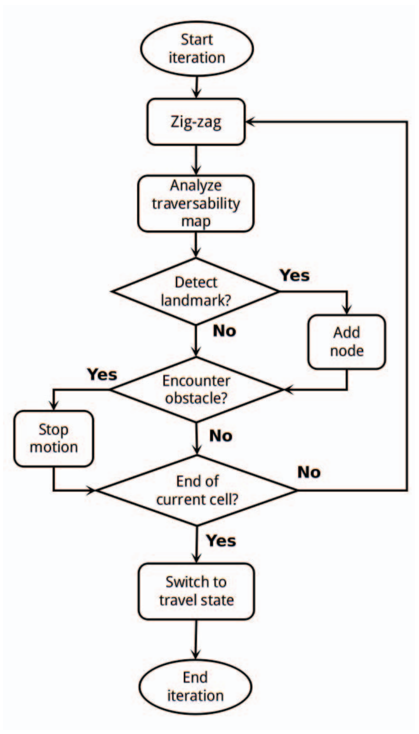


Fig. 3. Normal state algorithm.

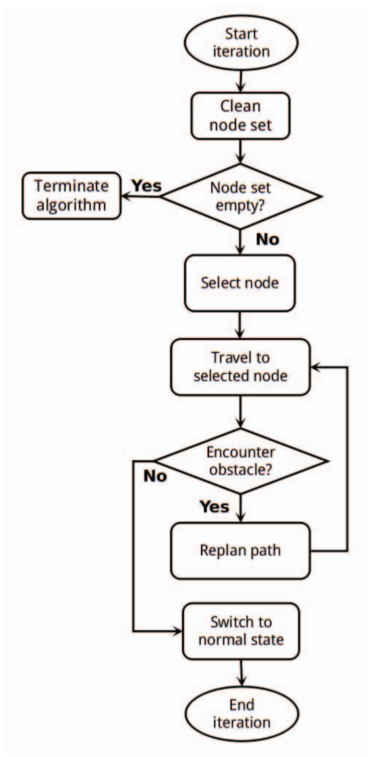


Fig. 4. Travel state algorithm.

surrounding area. When the robot reaches the end of the current cell (cf. Figure 10), it switches to *travel* state. In *travel* state, a node set cleaning is first performed to remove visited nodes in the node set. Then, the robot selects and travels to an unvisited node in the set. The path planning to the selected node can be accomplished either by a path planner, or by explicitly storing the connectivity between nodes. In our framework, a path planner designed specifically for legged robots [2] was used. Once the selected node is reached, the robot switches back to *normal* state. The algorithm terminates when there are no more unvisited nodes in the region of interest. Figure 2 shows a flow diagram of this two state iteration. Figures 3 and 4 present the *normal* state and the *travel* state algorithms, respectively.

B. Local Landmark Detection

1) Traversability Map Analysis: Landmark detection on the local level is accomplished by analyzing the robot-centric traversability map. The traversability map is divided into eight submaps (Figure 5). An obstacle detection algorithm is applied to each submap to get a binary label of obstacle occupancy, and the combination of these binary labels determines whether or not to add a node to the node set.

The obstacle detection algorithm consists of the following steps (cf. Figure 6). First, the submap is converted into a binary one by thresholding the traversability value. Subsequently, erosion and dilation are performed on the binary submap to reduce the impact of measurement noise. A contour finding is performed on the filtered submap, and the largest enclosed area among all contours is computed. If this area exceeds a certain threshold, the submap is classified as occupied by an obstacle. By filtering and thresholding the largest enclosed area, the algorithm is robust against measurement uncertainties. Moreover, it can be applied to arbitrarily shaped obstacles.

2) Nodes: Nodes are divided into two hierarchical categories: superior nodes and inferior nodes. In *travel* state, superior nodes have priority over inferior nodes to be selected as target to travel to. Figures 7 and 8 present the conditions for adding superior and inferior nodes. Nodes are added next to the robot with a lateral offset that is chosen such that the nodes fall into the respective side submap (labeled with 4 and 5 in Figure 7). By requiring the side submap to be free of obstacles, nodes are guaranteed to be within an area that

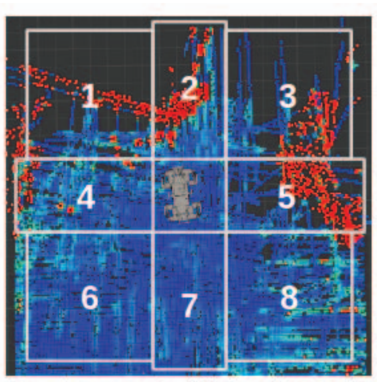


Fig. 5. A traversability map is divided into eight submaps for landmark detection. The color represents the traversability of the cells, with red being non-traversable and dark blue being perfectly traversable.

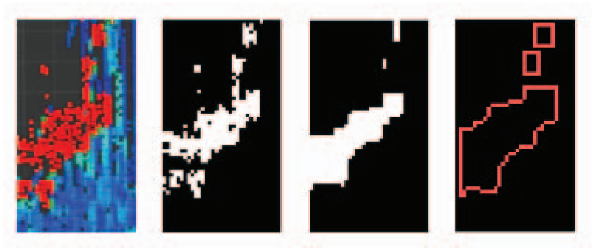
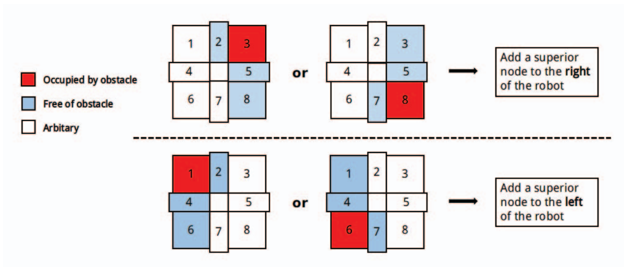
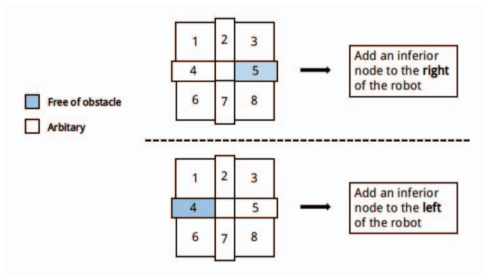


Fig. 6. Obstacle detection algorithm for a traversability submap. From left to right: original submap, binary thresholding of the traversability value, erosion and dilation, image contour.



Occupied by obstacle
Free of obstacle
Arbitary
1 2 3 4 5 6 7 8
or
Add a superior node to the right of the robot
Add a superior node to the left of the robot

Fig. 7. Traversability map condition for adding a superior node (robot moving in vertical direction).



Free of obstacle
Arbitary
1 2 3 4 5 6 7 8
Add an inferior node to the right of the robot
Add an inferior node to the left of the robot

Fig. 8. Traversability map condition for adding an inferior node (robot moving in vertical direction).

is reachable for the robot.

Superior nodes serve as connections to other cells. The adding condition captures the scenario where obstacles appear at the corner of robot's view, and rules out the scenario where the robot is walking along a wall, or is blocked by an obstacle (Figure 9). By going to a superior node, the robot reaches the corner of a potentially uncovered cell (Figure 10).

Inferior nodes serve as a backup when the robot failed to add a superior node, which could happen when the robot fails to detect obstacles in the corner area of the traversability map. A direct analogy can be drawn between inferior nodes and neighborhood grid cells used in approximate cellular decomposition approaches. Adding an inferior node to the side of the robot when no obstacle exists on the side is equivalent to checking unoccupied neighborhood grid cells in approximate cellular decomposition. The only difference is that the size and location of the effective grid cells in our approach vary and depend on the time when the new traversability map arrives. If all inferior nodes are visited, the algorithm guarantees that the region has been covered

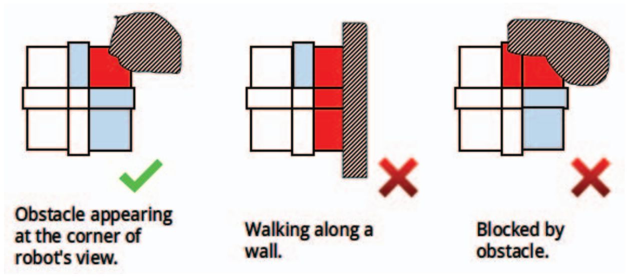


Fig. 9. Scenarios for adding a superior node (robot moving in vertical direction).

completely.

A node is marked as visited if it is selected as the target node during travel state. However, a node can also be visited by the robot while following the zig-zag pattern in *normal* state. Thus, a node set cleaning is required at the beginning of each travel state iteration to remove nodes that have been visited. This is accomplished by computing the uncovered fraction of an area centered at the node. If the uncovered fraction is smaller than a certain threshold, the node is classified as visited and removed from the node set. Proper tuning of the threshold is required. Lowering the threshold will reduce the time to finish coverage, at the cost of increasing the likelihood of incomplete coverage, and vice versa.

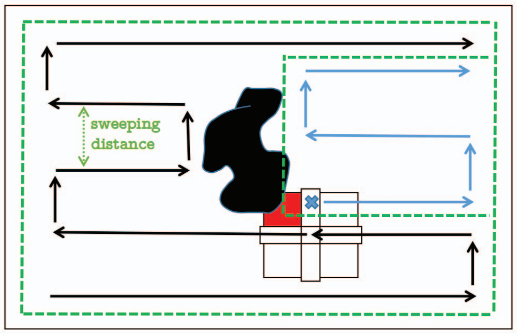


Fig. 10. An example of adding a superior node (shown as the blue cross mark). The dotted green line represents the approximate cell boundaries. Black and blue lines represent the zig-zag paths to cover the first and the second cell, respectively.

C. Obstacle Avoidance

Obstacle avoidance is also achieved using information from the traversability map. A safety submap is created by extracting data from a small part of the traversability map centered around a point that the robot will pass through (Figure 11), and this safety submap is analyzed using the obstacle detection algorithm mentioned earlier. Once the robot encounters an obstacle, the current motion is stopped. Then the robot either changes its moving direction according to the zig-zag pattern (*normal* state), or replans its path to the selected node (*travel* state).

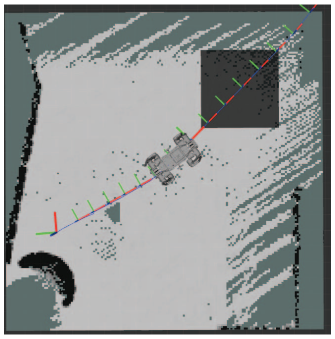


Fig. 11. The safety submap is shown as the dark square within the traversability map.

IV. Simulation

The proposed CPP algorithm was tested in two different experiments using *ROS* [19] and the open source robotics simulator *Gazebo* [20]. The first simulation was targeted at evaluating the algorithm in a flat terrain cluttered with obstacles and indoor-outdoor transitions. The second simulation aimed at assessing the performance in a rough, nonplanar terrain. A model of our quadruped *ANYmal* was used as robotic platform for the simulations. *ANYmal* is equipped with two rotating laser scanners (Hokuyo UTM-30LX-EW), one at the front and one at the back of the robot. From the sensor measurements, an elevation map and a traversability map [2], [21] of a robot-centric square area are built and updated with a certain frequency.

A. Planar Environment

In the first simulation, the robot was tasked with exploring a 14×13 m region (Figure 12), consisting of a partially confined area with three openings. In the simulation, the robot received traversability information of a 5×5 m square area centered at its position with a frequency of 0.5 Hz. The sweeping distance for zig-zag motion was set to 2 m (cf. Figure 10). Seven simulated experiments were performed with randomly selected starting points (cf. Figure 12). The results are listed in the following table.

Run ID	Covered area fraction (%)	Traveled length (m)	Covered area over traveled length (m)
1	99.2	137.05	1.32
2	97.6	154.86	1.15
3	98.9	121.99	1.48
4	99.2	118.13	1.53
5	99.4	125.06	1.45
6	99.1	117.10	1.54
7	98.2	154.42	1.16

The simulation results show that for all seven tests the robot was able to explore the region with at least 97.6% completeness. To evaluate the efficiency of coverage, we compute area covered per traveled distance, and the results are between 1.15 m and 1.54 m. For comparison, to cover an obstacle free region with the same size using a perfect zig-zag pattern, the robot would need to travel either 105 m or 111 m depending on the orientation of the zig-zag pattern, resulting in 1.73 m or 1.64 m of area covered per traveled distance.

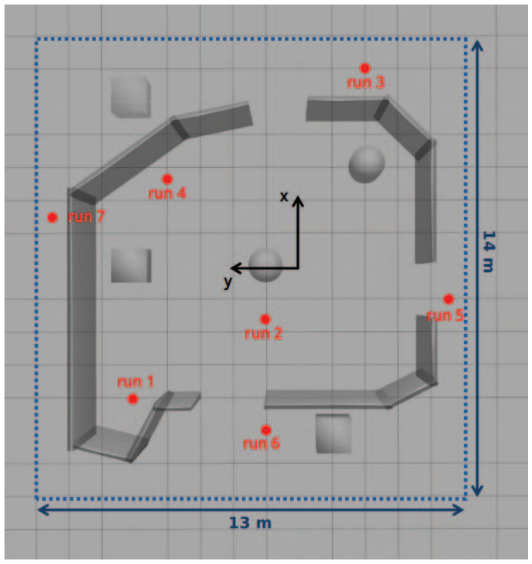


Fig. 12. Planar environment with walls and obstacles used for the first simulation.

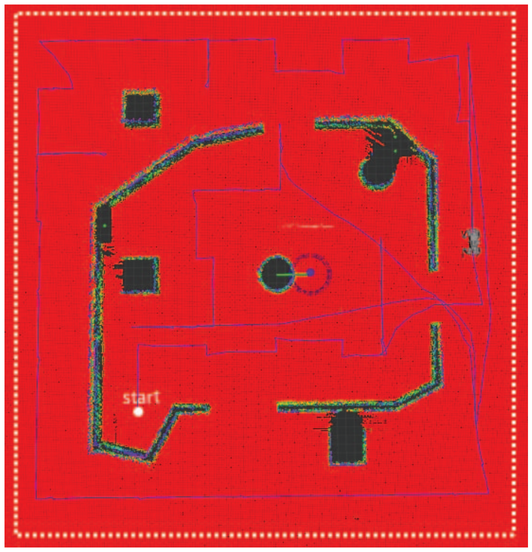


Fig. 13. Explored elevation map of run 1 in the first simulation. The blue line marks the traveled path of the robot, and the dotted white line marks the exploration boundary.

B. Rough Terrain

In the second simulation, the robot was tasked with exploring a 10×7 m region with rough terrain (Figure 14). Size and publishing frequency of the traversability map remained the same as in the first simulation. The obtained elevation map from the exploration is shown in Figure 15. To cover the region, the robot traveled 26.98 m. The simulation result shows that the proposed CPP algorithm can be used in rough terrain environments where obstacles cannot be easily detected from raw range sensor measurements.

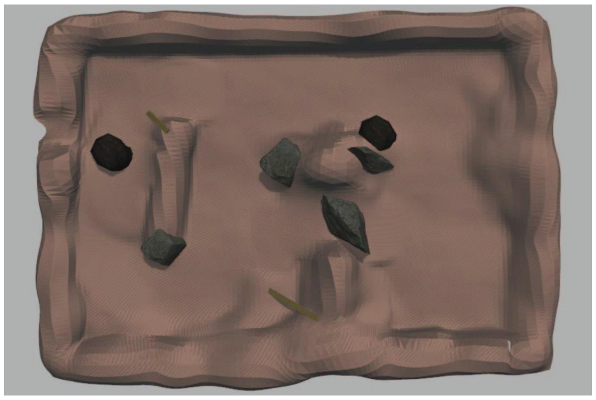


Fig. 14. Rough terrain environment used for the second simulation.

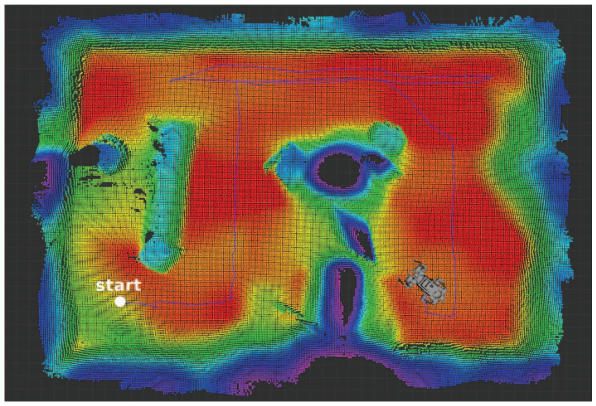


Fig. 15. Explored elevation map of the second simulation. The blue line marks the traveled path of the robot.

V. Conclusion and Future Work

In this paper, we developed a novel coverage path planning algorithm for guiding a legged robot to explore an initially unknown environment. The proposed algorithm uses exact cellular decomposition based on landmarks for planning on the global level, and approximate cellular decomposition for landmark detection in a local area within the sensor range. Traversability information of the surrounding area is used for landmark detection and obstacle avoidance. Landmark detection is robust to imperfect perception and can be applied to obstacles with any shape. Our algorithm improves over existing CPP approaches in several respects. First, it takes into account information about the traversability of the surrounding terrain, and robot's sensing range. Second, it allows starting the exploration from any point within the desired region, and it can be applied to any type of environment. Third, the algorithm is computationally inexpensive and does not utilize graph search. Two simulations of the proposed CPP algorithm were performed using the legged robot *ANYmal* as robotic platform. The results of the simulations demonstrate the feasibility of the algorithm for exploring environments with indoor-outdoor transitions, as well as rough terrain environments.

In the future, we plan to conduct experiments with the robot *ANYmal* in challenging outdoor and indoor environments. These experiments will include a detailed analysis of the impact of different terrain properties (terrain roughness, obstacle locations) on the performance of our CPP algorithm.

References

[1] M. Hutter, G. Christian, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, "ANYmal - A Highly Mobile and Dynamic Quadrupedal Robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016 (accepted for publication).

[2] M. Wermelinger, P. Fankhauser, R. Diethelm, P. Krüsi, R. Siegwart, and M. Hutter, "Navigation Planning for Legged Robots in Challenging Terrain," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016 (accepted for publication).

[3] H. Choset, "Coverage for robotics - A survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 113–126, 2001.

[4] E. Galceran and M. Carreras, "A Survey on Coverage Path Planning for Robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, December 2013.

[5] A. Zelinsky, R. Jarvis, J. C. Byrne, and S. Yuta, "Planning Paths of Complete Coverage of an Unstructured Environment by a Mobile Robot," in *International Conference on Advanced Robotics*, 1993.

[6] Y. Gabriely and E. Rimon, "Spiral-STC: an on-line coverage algorithm of grid environments by a mobile robot," in *IEEE International Conference on Robotics and Automation*, 2002, pp. 954–960.

[7] E. Gonzalez, O. Alvarez, Y. Diaz, C. Parra, and C. Bustacara, "BSA: A Complete Coverage Algorithm," in *IEEE International Conference on Robotics and Automation*, April 2005, pp. 2040–2044.

[8] M. Dakulović, S. Horvatić, and I. Petrović, "Complete Coverage D* Algorithm for Path Planning of a Floor-Cleaning Mobile Robot," in *the 18th International Federation of Automatic Control World Congress*, vol. 44, Jan 2011.

[9] S. Hert, S. Tiwari, and V. Lumelsky, "A terrain-covering algorithm for an AUV," *Autonomous Robots*, vol. 3, no. 2, pp. 91–119, June 1996.

[10] J. C. Latombe, *Robot Motion Planning*. Kluwer Academic, 1991.

[11] H. Choset and P. Pignon, "Coverage Path Planning: The Boustrophedon Cellular Decomposition," *Field and Service Robotics*, pp. 203–209, 1998.

[12] E. U. Acar, H. Choset, A. A. Rizzi, P. N. Atkar, and D. Hull, "Morse Decompositions for Coverage Tasks," *International Journal of Robotics Research*, vol. 21, no. 4, pp. 331–344, April 2002.

[13] E. U. Acar and H. Choset, "Sensor-based Coverage of Unknown Environments: Incremental Construction of Morse Decompositions," *International Journal of Robotics Research*, vol. 21, no. 4, pp. 345–366, April 2002.

[14] ——, "Robust sensor-based coverage of unstructured environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001, pp. 61–68.

[15] S. C. Wong and B. A. MacDonald, "A Topological Coverage Algorithm for Mobile Robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2003, pp. 1685–1690.

[16] ——, "Complete Coverage by Mobile Robots Using Slice Decomposition Based on Natural Landmarks," in *8th Pacific Rim International Conference on Artificial Intelligence*, August 2004, pp. 683–692.

[17] E. U. Acar, H. Choset, and P. N. Atkar, "Complete Sensor-based Coverage with Extended-range Detectors: A Hierarchical Decomposition in Terms of Critical Points and Voronoi Diagrams," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, November 2001, pp. 1305–1311.

[18] E. U. Acar, H. Choset, and J. Y. Lee, "Sensor-Based Coverage With Extended Range Detectors," in *IEEE Transactions on Robotics*, vol. 22, no. 1. IEEE, February 2006, pp. 189–198.

[19] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[20] K. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2004, pp. 2149–2154.

[21] P. Fankhauser and M. Hutter, "A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation," in *Robot Operating System (ROS)—The Complete Reference (Volume 1)*, A. Koubaa, Ed. Springer, 2016, ch. 5.