FT-MSTC*: An Efficient Fault Tolerance Algorithm for Multi-robot Coverage Path Planning

Chun Sun, Jingtao Tang and Xinyu Zhang

Abstract— **Fault tolerance is very important for multi-robot systems, especially for those operated in remote environments. The ability to tolerate failures, allows robots effectively to continue performing tasks without the need for immediate human intervention. In this paper, we present a new efficient fault tolerance algorithm for multi-robot coverage path planning (mCPP). The entire coverage path is considered as a topological task loop. The ideal mCPP problem is handled by partitioning this task loop and assign each partition to individual robot. When a faulty robot is detected, we use an optimization method to minimize the overall maximum coverage cost while considering both the tasks accomplished before robot failures and the remaining tasks. We perform various experiments for regular grid maps and real field terrains. We compare our algorithm against other coverage path planning algorithms and our algorithm outperforms existing spiral-STC-based methods in terms of the overall maximum coverage cost.**

I. INTRODUCTION

Coverage path planning (CPP) is the problem of determining a set of paths that cover the area of interest while avoiding obstacles[1]. Coverage path planning has many indoor and outdoor robotic applications, such as floor cleaning robots[2], autonomous underwater vehicles[3], unmanned aerial vehicles[4], demining robots[5], automated harvesters[6], planetary exploration[7], search and rescue operations[8], and lawn mowers[9].

For large-scale tasks, a multi-robot system can significantly improve the efficiency of coverage path planning using workload distribution and efficient collaboration. A multi-robot system reduces the need for massive humans to perform tedious jobs. In addition, a multi-robot system can improve robustness in case of failure of some robots. Faulty robots can cause serious damages in a robotic system and economic loss[10], [11]. Therefore, fault tolerance is very important for multi-robot systems, especially those operated in remote environments. For example, our research was developed for a project of tree planting robots to restore vast degraded lands. These robots work in remote and hazardous environments, where land terrains exhibit complex surface, various obstacles and topology. The ability to tolerate failures, allows robots effectively continue performing tasks without the need for immediate human intervention. When a robot encounters failures, other robots need to accomplish the remaining tasks. There are a few early work addressing the problem of fault tolerance for multi-robot systems using centralization[10], [12] and self-organization[13], [14]. Moreover, many of these work focused on fault detection. It is desirable to design an efficient fault tolerance algorithm for re-planning multi-robot coverage paths.

The authors are with Shanghai Key Laboratory of Trustworthy Computing, Engineering Research Center of Software/Hardware Co-design Technology and Application (MoE) and School of Software Engineering, East China Normal University, Shanghai. Xinyu Zhang is the corresponding author. E-mail: xyzhang@sei.ecnu.edu.cn.

Main Results: In this paper, we present a new efficient fault tolerance algorithm for multi-robot coverage path planning using optimization method. This work is an extension to MSTC*[15] by incorporating fault tolerance. We treat multi-robot coverage path planning (mCPP) as the problem of partitioning a topological task loop and assign each task partition to individual robot. To re-plan a set of well-balanced tasks after failures, we minimize the overall maximum coverage cost by considering both the tasks accomplished before robot failures and the remaining tasks. We compare our algorithm against other coverage path planning methods, including a naïve algorithm that assigns the unfinished tasks by the faulty robot to the fastest one, and the entirely re-planning algorithm. We perform a few experiments for regular grid maps and field terrains for comparison. The experimental results show that our algorithm outperforms existing spiral-STC-based methods in terms of the overall maximum coverage cost.

II. RELATED WORK

Here, we briefly review the work directly relevant to fault tolerance for multi-robot systems.

Decentralization is considered fault-tolerant and scalable since a decentralized multi-robot system has distributed architecture and modular[16]. The work in [10] used a decentralized solution to re-scout and cover the regions with a group of UAVs. Since the UAVs always follow a certain flock, it is less efficient for a large-scale coverage. Viet et al. [17] proposed an online multi-robot coverage system using only local interactions to coordinate and construct simultaneously non-overlapping regions in an incremental manner. This algorithm merely deals with 2D environments. Recently, Hyatt et al. [18] introduced a multi-robot online CPP algorithm based on Monte Carlo Tree Search (MCTS). The algorithm allows robots to cover different non-convex areas, but it often has low efficiency and high path overlapping ratio. Gavran et al. [19] presented a multi-robot task server with fault tolerance, allowing multiple robots work together using constraint-based formulation for simultaneous task assignment and plan generation. It provides a scalable and robust infrastructure for multi-robot programming.

In a centralized system, Jain et al. [11] proposed a fault-tolerant coverage method in a limited communication

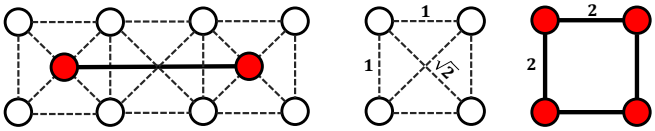
range scenario and used Voronoi partitions for coverage distribution. The region exhibiting faulty robots is handled by the neighbouring agents. The work in [12] introduced a reduced constrained Delaunay triangulation to build a terrain graph, and used supportive tree to cover the failure paths. Jamshidpey et al. [14] proposed a mergeable nervous system and used a fault tolerance solution with self-organized communication topologies [13]. Luo et al. [20] presented a collective coverage control strategy for large-scale spilling scenarios. They isolated the faulty robots and re-planned the coverage tasks until the robots regained its ability. Li et al. [21] proposed a fault-tolerant solution for multi-robot patrolling tasks using a dynamic priority queue for the turn taking and time-out re-planning mechanism. Many work focuses on fault detection. Instead, our work aims to improve the efficiency of coverage path re-planning for a centralized multi-robot coverage path planning system.

III. Preliminaries & Problem Definition

Given multiple (k) robots and a terrain, the goal of mCPP is to cover the given terrain using these robots. According to the tasks, the coverage terrain is often divided into a large number of cells and the cell dimensions depend on specific applications. For example, in our project for tree planting robots, the spacing between the lines in plantation is five meters and the spacing of plants within a line is three meters[15]. Here, we first introduce some preliminaries, notation and problem definition.

A. Notation

We represent the terrain (cells) as a graph, namely *covering graph*, denoted by $\mathcal{G}$ (see Figure 1-(a)). The nodes in $\mathcal{G}$ is covering nodes, denoted by π. Two adjacent nodes are connected by an edge e. In spiral-STC based algorithms [22], a spanning graph is used to efficiently generate coverage paths. Here, we denote a spanning graph by $\mathcal{H}$ and its nodes are *spanning nodes*. Note that a covering node is associated to merely one spanning node. Both $\mathcal{G}$ and $\mathcal{H}$ are edge-weighted graphs. The edge weights $\|e\|$ are shown in Figure 1-(b) and (c).



(a) Covering Graph $\mathcal{G}$ & Spanning Graph $\mathcal{H}$ (b) $\|e\|$ in $\mathcal{G}$ (c) $\|e\|$ in $\mathcal{H}$

Fig. 1. Covering graph $\mathcal{G}$ is represented by white nodes and dotted edges, and spanning graph $\mathcal{H}$ is represented by red nodes and solid edges. (a) A spanning node (red) is generated from four adjacent covering nodes (white); (b) A weight (1 or $\sqrt{2}$ is assigned to the edges of $\mathcal{G}$; (c) A weight 2 is assigned to the edges of $\mathcal{H}$.

B. mCPP Problem Definition

Given $\mathcal{G}$, $\mathcal{H}$ and k robots, a coverage path Π_i to be travelled by robot $\mathcal{R}_i$ is a set of nodes $\{\pi_i^j\} \in \mathcal{H}$ (see Figure 2-(a)). Its accumulating weight (i.e. cost) is denoted by $\mathcal{W}_i$. Ideally, all the robots will operate successfully and we aim at computing a set of coverage paths $\{\Pi_1, \Pi_2, \ldots, \Pi_k\}$ for k robots through minimizing the maximum of $\mathcal{W}_{\Pi_i}$. That is

$$\arg\min_{\{\Pi_i\}} \left(\max_{1 \le i \le k} (\mathcal{W}_{\Pi_i}) \right). \quad (1)$$

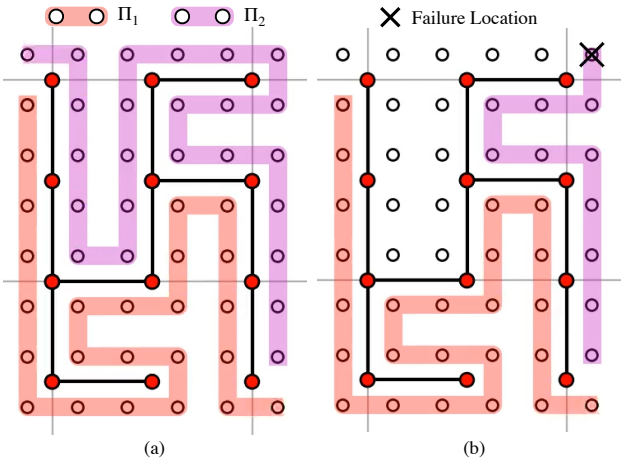


Fig. 2. Multi-robot coverage path planning with and without faulty robots. (a) Two coverage paths Π_1 and Π_2 are generated without faulty robots. (b) A robot II stops working due to hardware/software failures. Its failure location is highlighted by $\times$. Other robots are required to continue performing its remaining tasks without the need for immediate human intervention.

C. Fault Tolerance mCPP Pipeline

Our fault tolerance algorithm for mCPP consists of two stages.

1) We first use MSTC* algorithm [15] to plan a set of coverage paths $\{\Pi_i\}$. Ideally, $\{\Pi_i\}$ will be travelled by associating robots $\{\mathcal{R}_i\}$.
2) If a faulty robot is detected, we re-plan a set of well-balanced tasks by minimizing the overall maximum coverage cost while considering both the tasks accomplished before robot failures and the remaining tasks.

IV. Fault Tolerance Algorithm for mCPP

In this section, we introduce our fault tolerance algorithm for multi-robot coverage path planning. For a given covering graph $\mathcal{G}$ and the associating spanning graph $\mathcal{H}$, we first use MSTC* algorithm[15] to generate a circumnavigating coverage path in $\mathcal{G}$ by following the right-side of the traversal route in $\mathcal{H}$. The coverage path can be considered as an end-to-end loop Π (see Figure 3). For multi-robot coverage path planning, the problem is solved by dividing the loop evenly using a set of proper partition node π.

When a faulty robot is detected, a straightforward solution is to use an additional robot to replace it and continue performing the remaining tasks. However, in remote and hazardous environment, it is often costly and even infeasible. Another straightforward method is to assign the remaining tasks of the faulty robot to its nearby robot or the fastest robot (i.e., Naïve-MSTC). Or a Balanced-MSTC algorithm

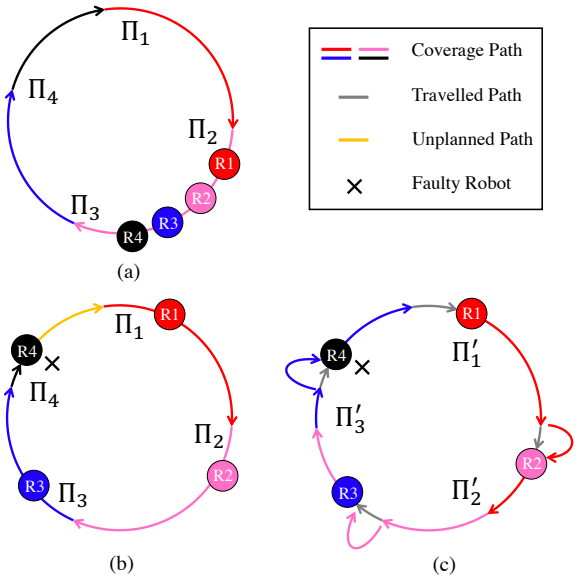


Fig. 3. Our coverage path partition strategy. (a) An even partition; (b) $\mathcal{R}_4$ fails to operate; (c) A new loop is generated and connected by shortcut paths (represented by curves).

can be used to entirely re-plan the remaining paths for k-1 robots.

Though these straightforward solutions mentioned above can be directly used, we propose a more practical and efficient solution to generate the evenly distributed tasks for all robots by taking account of historic tasks and remaining tasks.

A planned coverage path Π_i can be divided into two components. The one is the path travelled before robot failure (Π_i^{BF}) and the other is the remaining path to be travelled in future (Π_i^{AF}). Without loss of generality, we assume robot $\mathcal{R}_k$ is faulty. In order to cover the remaining path Π_i^{AF}, we compute a new set of coverage paths $\{\Pi'_1, \Pi'_2, \ldots, \Pi'_{k-1}\}$ for remaining k-1 robots. We consider both coverage costs of the travelled path $\mathcal{W}_{\Pi_i^{\text{BF}}}$ and the new path $\mathcal{W}_{\Pi'_i}$ to make the coverage more efficient. In practice, some extra costs have to be spent on bypassing those travelled nodes since travelling costs cannot be neglected for large-scale tasks. We denote the bypassing costs as $\mathcal{W}_{\Pi'_{i,\text{bypass}}}$. Then we aim to minimize the following accumulating cost

$$\arg\min_{\{\Pi'_i\}} \left(\max_{1 \le i \le k-1} \left(\mathcal{W}_{\Pi_i^{\text{BF}}} + \mathcal{W}_{\Pi'_i} + \mathcal{W}_{\Pi'_{i,\text{bypass}}} \right) \right). \quad (2)$$

Figure 3 gives an example to illustrate the process. For given four robots, the end-to-end circular loop Π is first divided into four balanced paths Π_i(i=1,2,3,4) and ideally, Π_i is handled by robot $\mathcal{R}_i$ (see Figure 3-(a)). Here, we assume robot $\mathcal{R}_4$ fails, there will be four travelled paths Π_i^{BF} and four remaining paths Π_i^{AF} (see Figure 3-(b)). Our goal is to find the optimal task partitions for the rest working robots.We elaborate our FT-MSTC* algorithm as follows.

When a faulty robot is detected, each planned coverage path Π_i is divided into: travelled path (Π_i^{BF}) and remaining path (Π_i^{AF}). To plan a set of new coverage paths for k-1 robots, we connect the remaining paths and distribute them to the k-1 robots.

We first create a shortcut path for each travelled path $\Pi'_{i,\text{BF}}$, which connects the beginning node and the ending node of Π_i^{BF} (see Figure 3-(c)). We use the classic A* pathfinding algorithm to generate these shortcut paths and assigned a bypassing cost $\mathcal{W}_{\Pi'_{i,\text{bypass}}}$ to individual shortcut path. Meanwhile, these shortcut path and the remaining paths Π_i^{AF} will construct a new end-to-end loop Π' (see Figure 3-(c))).

Then, for k-1 robots, a new set of paths $\{\Pi'_i\}(i = 1, 2, \ldots, k-1)$ will be generated iteratively. The main idea is to, starting with the initial rough partitions, generate balanced workload for all robots by iteratively minimizing the maximum weights (refer to Eq. (2)). The coverage path with the maximum weight is denoted by Π'_{max} and the coverage path with the minimum weight is denoted by Π'_{min}. If Π'_{max} and Π'_{min} are adjacent, we iteratively remove the nodes at their boundary from Π'_{max} and append them into Π'_{min} until they are balanced. However, if Π'_{max} and Π'_{min} are not adjacent. We gradually shift the nodes from Π'_{max} to Π'_{min} through in-between partitions. During the shifting, the workload of the in-between partitions remain unchanged.

Finding the best partitions is a NP-hard problem. However, our algorithm uses a greedy strategy to gradually approximate the optimal partitions.

V. Results & Analysis

In this section, we present some implementation details of our fault tolerance algorithm, and demonstrate some experimental results and comparison.

A. Regular Grid Maps

We first test our algorithm using small regular grid maps. As shown in Figure 4, each robot starts from its individual depot. The robot depots represented with $*$ are located at the bottom. These robot depots are grouped and highlighted using rounded rectangles. A 5×10 unweighted map shown in Figures 4-(a) and (e), was used in some previous work [23], [24], [25]. Other regular grid maps in Figure 4 are constructed using random weights. The obstacles (blocked cells) are represented using symbols $\times$. In our experiment, we allow one robot stop working after moving 20 nodes. The failure locations are highlighted using red circles. Figures 4-(a)$\sim$(d) show the snapshot when a faulty robot is detected. Figures 4-(e)$\sim$(h) demonstrate the corresponding results of coverage path planning using our FT-MSTC* algorithm.

Comparison: Figure 5 shows the performance of our FT-MSTC* algorithm. We also compare our algorithm in terms of the maximum coverage cost against a straightforward algorithm (denoted by Naïve-MSTC, an algorithm of assigning the tasks unfinished by the faulty robot to the fastest one) and MSTC algorithm (denoted by Balanced-MSTC, an entire re-planning algorithm). Note that, the lower the maximum coverage cost, the better the performance. The experimental

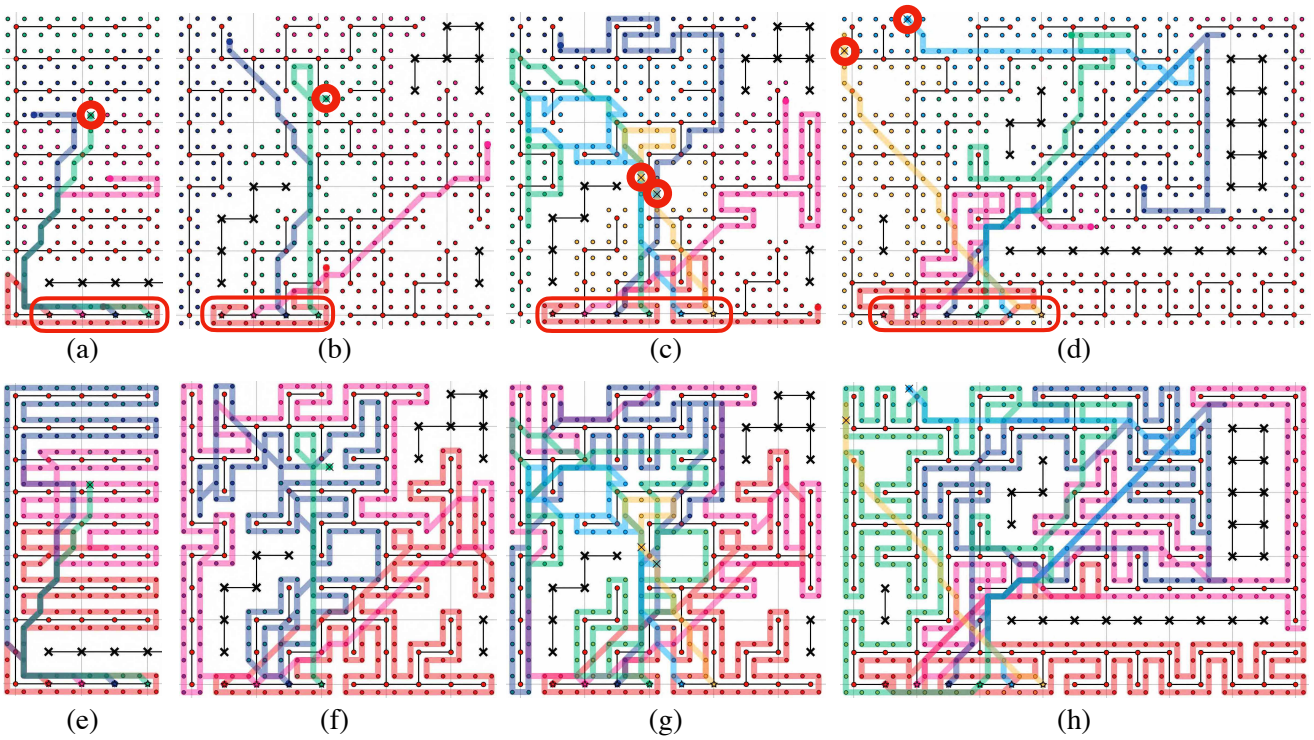


Fig. 4. Regular Grid Terrains. Robot's failure location is highlighted using red circles. (a)∼(d): The snapshot when a faulty robot is detected. (e)∼(h): The corresponding results of coverage path planning using our FT-MSTC* algorithm. Four robots are used in (a) and (b). Six robots are used in (c) and (d). A faulty robot is detected after it moves 20 nodes.

results show that our FT-MSTC* algorithm outperforms these two methods (see Figure 5).

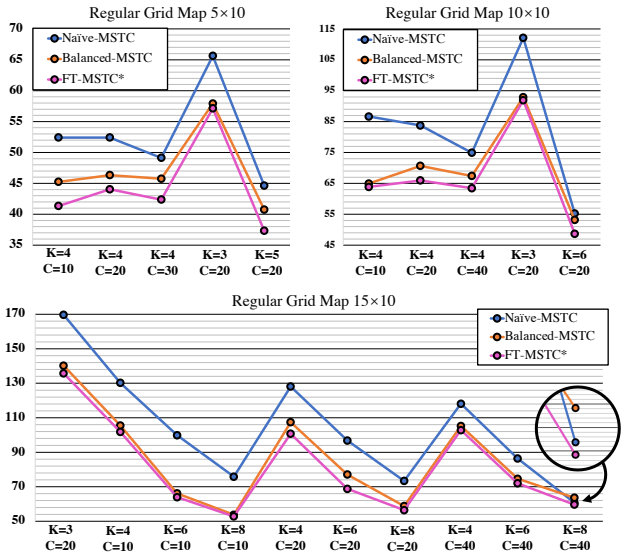


Fig. 5. Performance and scalability comparison for regular grid maps in terms of the maximum weights of coverage paths. The K refers to the robot number and C refers to the steps when faulty robots are detected. Our FT-MSTC* algorithm outperforms the other two algorithms.

B. Field Terrains

We also applied our algorithm to field terrains in real world. We divide the field terrains into 256×256 cells. Before generating the coverage result, we first apply a traversability analysis to the satellite map to remove the unreachable cells from the map.

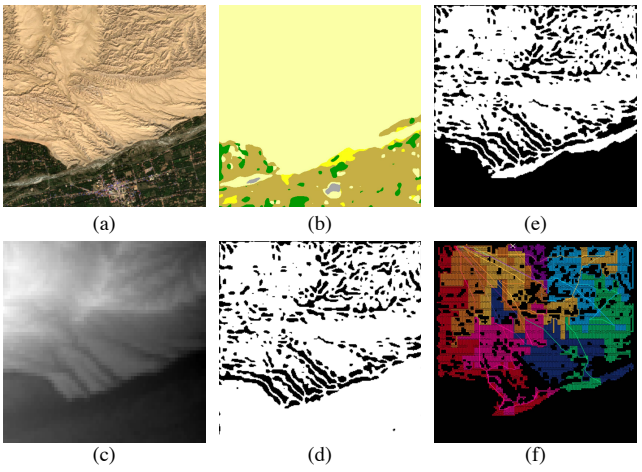


Fig. 6. Traversability analysis of a field terrain around (78.48°E, 37.22°N).(a) Satellite map image. (b) Land cover predict. (c) DEM representing the height information. (d) DEM filtering reulst. (e) The fusion result of land cover predict and DEM filtering. (f) The coverage path planning on the terrain.

We train a DeepLab [26] network using the SEN12MS

and DFC2020 dataset [27], [28], and predict a pixel-wised land cover classification on a given Sentinel-2 satellite image [29]. Then, a DEM filter is applied to the SRTM height map [30]. The slope greater than a threshold will be marked as unworkable regions (see Figure 6). In our application, the target work areas are the barren regions. Therefore, we generate a workable traversability map by combining the barren data on Figure 6-(b) and the traversability regions on Figure 6-(d). After traversability analysis, some unreachable nodes and edges are removed from graph. For example, 10861 nodes and 19919 edges remain in the field terrain shown in Figure 7-(a). 8665 nodes and 15548 edges are left in Figure 7-(b). 8675 nodes and 15460 edges are left in Figure 7-(c). 10238 nodes and 17931 edges are left in Figure 7-(d).

Comparison: Figure 8 shows the performance of three algorithms on the field terrains. As shown in Figure 8-(a) and (b), our FT-MSTC* shows better performance than the other algorithms. It also shows that our FT-MSTC* can efficiently reduce the loss when failures happen in early stage.

In addition, we evaluate the performance while changing failure time instance and location. The results are shown in Figure 9. It is obvious that the maximum coverage weights reduce when the failure time instance increases. Our FT-MSTC* outperforms Balanced-MSTC.

VI. CONCLUSIONS

We have presented an efficient fault tolerance algorithm (FT-MSTC*) for multi-robot coverage path planning. Our experiments for regular grid maps and field terrains show superior performance against other coverage path planning algorithms in terms of the overall maximum coverage cost.

There are a few limitations in our algorithm. First, our algorithm follows a greedy strategy and there is no guarantee to find the best partition result. Second, our algorithm becomes less efficient when there are many faulty robots. The extra costs spent on bypassing the already travelled nodes may increase dramatically.

For future work, we would like to improve the performance using centralized reinforcement learning or multi-agent reinforcement learning. Our algorithm is designed for centralized multi-robot systems. We would like to extend our algorithm to handle decentralized multi-robot systems.

REFERENCES

[1] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[2] F. Yasutomi, M. Yamada, and K. Tsukamoto, "Cleaning robot control," in *IEEE International Conference on Robotics and Automation*, 1988, pp. 1839–1841.

[3] S. Hert, S. Tiwari, and V. Lumelsky, "A terrain-covering algorithm for an AUV," in *Underwater Robots*. Springer, 1996, pp. 17–45.

[4] A. Ahmadzadeh, J. Keller, G. Pappas, A. Jadbabaie, and V. Kumar, "An optimization-based approach to time-critical cooperative surveillance and coverage with UAVs," in *Experimental Robotics*, 2008, pp. 491–500.

[5] E. U. Acar, H. Choset, Y. Zhang, and M. Schervish, "Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods," *International Journal of Robotics Research*, vol. 22, no. 7-8, pp. 441–466, 2003.

[6] M. Ollis and A. Stentz, "Vision-based perception for an automated harvester," in *IEEE/RSJ International Conference on Intelligent Robot and Systems*, 1997, pp. 1838–1844.

[7] L. Heng, A. Gotovos, A. Krause, and M. Pollefeys, "Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 1071–1078.

[8] L. Lin and M. A. Goodrich, "UAV intelligent path planning for wilderness search and rescue," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 709–714.

[9] M. Bosse, N. Nourani-Vatani, and J. Roberts, "Coverage algorithms for an under-actuated car-like vehicle in an uncertain environment," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 698–703.

[10] M. De Benedetti, F. D'Urso, G. Fortino, F. Messina, G. Pappalardo, and C. Santoro, "A fault-tolerant self-organizing flocking approach for UAV aerial survey," *Journal of Network and Computer Applications*, vol. 96, pp. 14–30, 2017.

[11] R. Jain, R. Tiwari, P. Jain, and P. Sujit, "Distributed fault tolerant and balanced multi-robot area partitioning for coverage applications," in *IEEE International Conference on Unmanned Aircraft Systems*, 2018, pp. 293–299.

[12] P. Fazli, A. Davoodi, P. Pasquier, and A. K. Mackworth, "Fault-tolerant multi-robot area coverage with limited visibility," in *IEEE International Conference on Robotics and Automation*, 2010.

[13] W. Zhu, M. Allwright, M. K. Heinrich, S. Oğuz, A. L. Christensen, and M. Dorigo, "Formation control of UAVs and mobile robots using self-organized communication topologies," in *International Conference on Swarm Intelligence*. Springer, 2020, pp. 306–314.

[14] A. Jamshidpey, W. Zhu, M. Wahby, M. Allwright, M. K. Heinrich, and M. Dorigo, "Multi-robot coverage using self-organized networks for central coordination," in *International Conference on Swarm Intelligence*. Springer, 2020, pp. 216–228.

[15] J. Tang, C. Sun, and X. Zhang, "MSTC*:multi-robot coverage path planning under physical constraints," in *IEEE International Conference on Robotics and Automation*, 2021, pp. 1–6.

[16] R. Almadhoun, T. Taha, L. Seneviratne, and Y. Zweiri, "A survey on multi-robot coverage path planning for model reconstruction and mapping," *SN Applied Sciences*, vol. 1, 07 2019.

[17] H. H. Viet, V.-H. Dang, S. Choi, and T. C. Chung, "Bob: an online coverage approach for multi-robot systems," *Applied Intelligence*, vol. 42, no. 2, pp. 157–173, 2015.

[18] P. Hyatt, Z. Brock, and M. D. Killpack, "A versatile multi-robot monte carlo tree search planner for on-line coverage path planning," 2020. [Online]. Available: https://arxiv.org/abs/2002.04517

[19] I. Gavran, R. Majumdar, and I. Saha, "Antlab: a multi-robot task server," *ACM Transactions on Embedded Computing Systems*, vol. 16, no. 5, pp. 1–19, 2017.

[20] S. Luo, J. H. Bae, and B. Min, "Pivot-based collective coverage control with a multi-robot team," in *IEEE International Conference on Robotics and Biomimetics*, 2018, pp. 2367–2372.

[21] N. Li, M. Li, Y. Wang, D. Huang, and W. Yi, "Fault-tolerant and self-adaptive market-based coordination using hoplites framework for multi-robot patrolling tasks," in *IEEE International Conference on Real-time Computing and Robotics*, 2018, pp. 514–519.

[22] 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*, vol. 1, 2002, pp. 954–960.

[23] Xiaoming Zheng, Sonal Jain, S. Koenig, and D. Kempe, "Multi-robot forest coverage," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 3852–3857.

[24] Xiaoming Zheng and S. Koenig, "Robot coverage of terrain with non-uniform traversability," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 3757–3764.

[25] X. Zheng, S. Koenig, D. Kempe, and S. Jain, "Multirobot forest coverage for weighted and unweighted terrain," *IEEE Transactions on Robotics*, vol. 26, no. 6, pp. 1018–1031, 2010.

[26] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *European conference on computer vision*, 2018, pp. 801–818.

[27] M. Schmitt, L. H. Hughes, C. Qiu, and X. X. Zhu, "SEN12MS - A curated dataset of georeferenced multi-spectral sentinel-1/2 imagery for deep learning and data fusion," *CoRR*, vol. abs/1906.07789, 2019. [Online]. Available: http://arxiv.org/abs/1906.07789

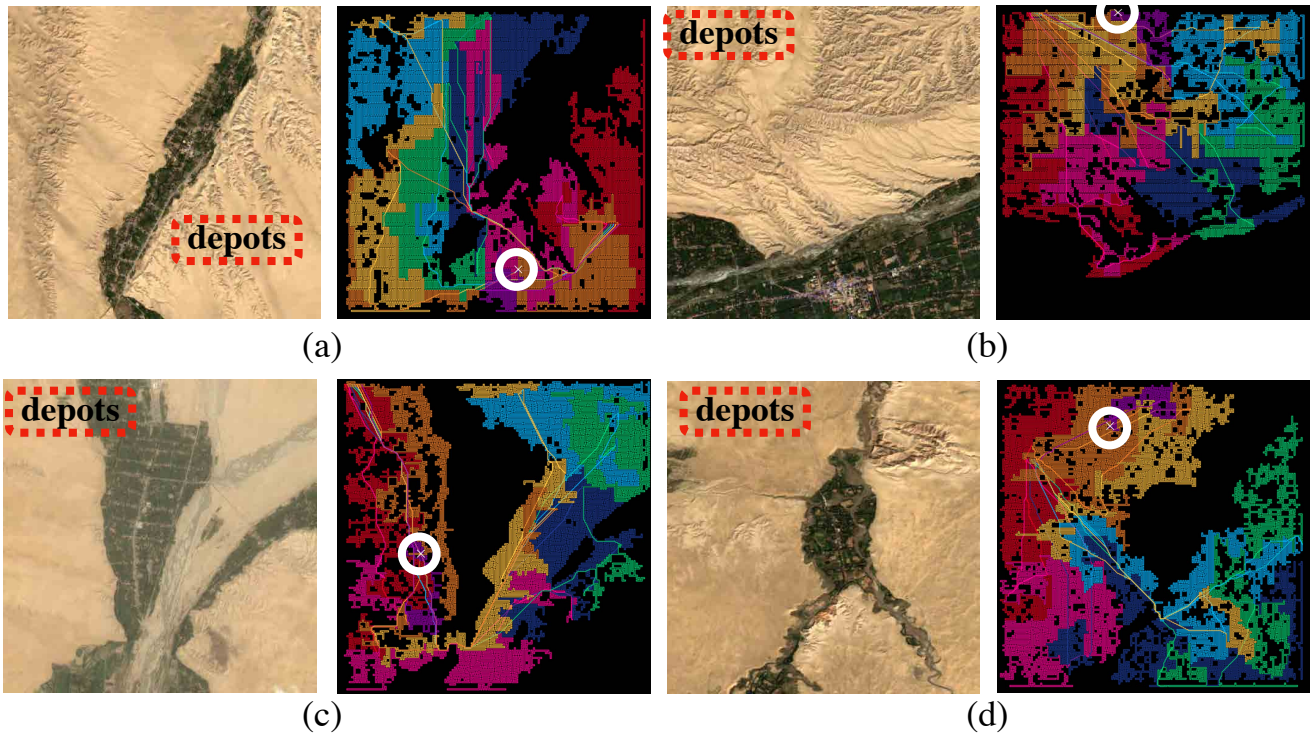


Fig. 7. Real Field Terrains. The robot depots are highlighted with red rectangles and the failure locations are represented using white circles. There are $k = 8$ robots and faulty robots are detected at time step 1000).

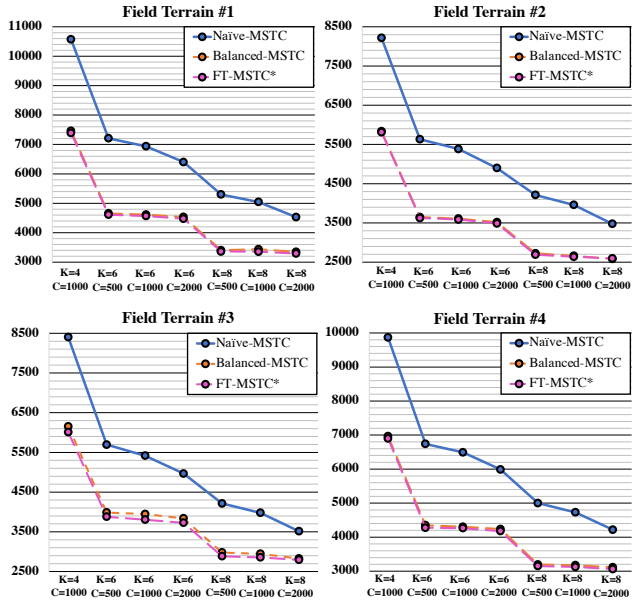


Fig. 8. Performance and scalability comparison for field terrains in terms of the maximum weights of coverage paths.

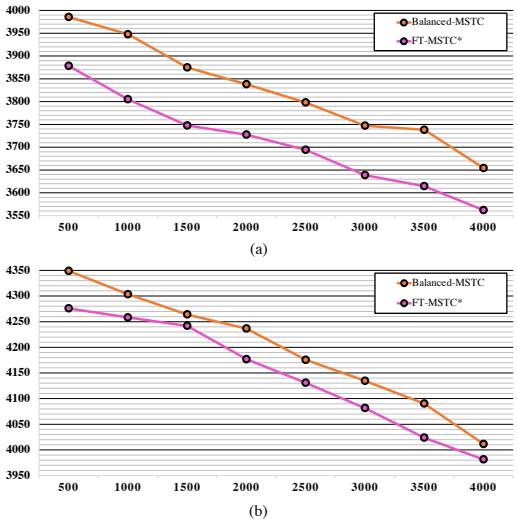


Fig. 9. Comparison between our FT-MSTC* and Balanced-MSTC in terms of the maximum coverage weights when varying failure time step from 500 to 4000 ($k = 6$). (a) Real field terrain #1 and (b) Real field terrain #2. Our FT-MSTC* outperforms Balanced-MSTC algorithm.

[28] H. Michael, Schmittand Lloyd, G. Pedram, Y. Naoto, and H. Ronny, "IEEE GRSS data fusion contest," 2020. [Online]. Available: http://dx.doi.org/10.21227/rha7-m332

[29] M. Drusch, U. Del Bello, S. Carlier, O. Colin, V. Fernandez, F. Gascon, B. Hoersch, C. Isola, P. Laberinti, P. Martimort, *et al.*, "Sentinel-2: ESA's optical high-resolution mission for GMES operational services," *Remote sensing of Environment*, vol. 120, pp. 25–36, 2012.

[30] A. Jarvis, H. I. Reuter, A. Nelson, E. Guevara, *et al.*, "Hole-filled SRTM for the globe version 4," *available from the CGIAR-CSI SRTM 90m Database (http://srtm. csi. cgiar. org)*, vol. 15, pp. 25–54, 2008.