EasyChair Preprint

№ 1460

Multi-Rover Exploration Strategies: Coverage Path Planning with Myopic Sensing

Mickaël Laîné and Kazuya Yoshida

September 4, 2019

Multi-Rover Exploration Strategies: Coverage Path Planning with Myopic Sensing

Mickaël Laîné, Kazuya Yoshida

Abstract The past few decades have seen tremendous progress in planetary exploration rovers and the most groundbreaking missions in space exploration history. Although the main focus has been toward single rover deployment, there has also been substantial research into swarm intelligence and multi-rover systems. This paper presents single and multi-rover path planning strategies for future planetary exploration missions. The main objective is to develop real-time Coverage Path Planning (CPP) solutions in unknown environments. The algorithms presented here are based on simple building blocks that use solely myopic sensing information to iteratively compute a motion cost function and decide their next move. Several single rover exploration methods are introduced and compared through simulation. The results show that complete coverage is not always possible in planetary exploration scenarios. Also, by slightly reducing the required percent of terrain to explore, the efficiency can be improved. In addition to this, two variants of multi-rover path planning algorithms are defined. A first method is a fully collaborative method where the rovers explore a region together and a second where the terrain is segmented and the rover operate separately from each other.

Laîné, Mickaël
Dept. of Aerospace Engineering, Tohoku University, Sendai, JAPAN e-mail: mickael@astro.mech.tohoku.ac.jp

Yoshida, Kazuya
Dept. of Aerospace Engineering, Tohoku University, Sendai, JAPAN e-mail: yoshida@astro.mech.tohoku.ac.jp

1 Introduction

1.1 Background and Motivations

The future of planetary exploration is tightly related to multi-rover collaboration and swarm intelligence. This implies that groups of autonomous rovers will work together to complete their mission. Soon, scenes like the one depicted in Fig.1, could become a mainstream feature of the lunar landscape. With teleoperation difficult for large groups of rovers, not only due to the time delay in communication, autonomous software is essential. Possible missions could be scientific analysis of vast regions, resource prospecting or scouting locations for future human settlements. In any case, one of the vital systems is an efficient path planning method that has to be tailored depending on rover hardware and mission objectives. Taking lunar surface prospecting as an example, the sensor dedicated to resource detection will most likely require ground sampling and analysis. The path planning algorithm must allow the rover to visit most if not all of the environment. To do so, algorithms known as Coverage Path Planning (CPP) must be used. Relying on GPS and global mapping, it is relatively easy to develop path planning and navigation solutions. In reality, this type of information is not expected to be available in planetary exploration scenarios.

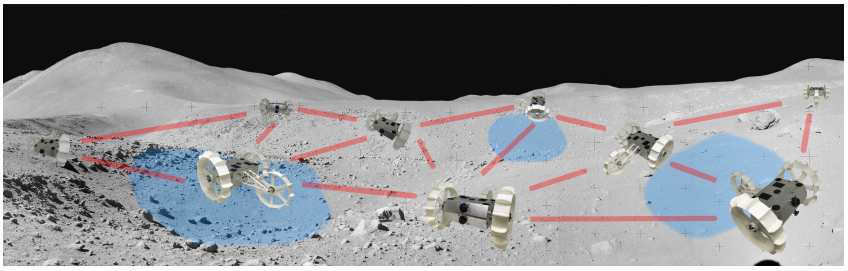


Fig. 1: A representation of inter-connected micro-rovers performing resource prospecting in a lunar scenery.

Another point of consideration is related to the internal hardware of the rovers. Both large planetary rovers like NASA's Curiosity and lighter micro-rovers [16], such as the Koguma platform [9] shown in Fig.2, have severe limitations in computation power. Autonomous rovers require robust software to ensure they function properly throughout their lifetime and recover from potential failures. Probably the most important system is Simultaneous Localisation and Mapping (SLAM). Knowing the location of the rover for navigation and mission coordination is important, but when prospecting for resources, having accurate locations of ground sampling is essential. However, SLAM is computationally demanding, as seen in [15], where they benchmark the requirements and efficiency of some well known algorithms.

Adding the needs of SLAM and all other necessary systems, the path planning solution needs to be able to work in real-time with minimum computation requirements. Micro-rovers are also limited by their sensing capabilities and can only perceive their surroundings within a limited range. Developing path planning solutions relying solely on this myopic information is challenging and has currently not been well studied.

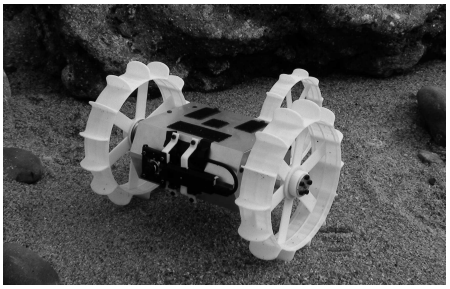


Fig. 2: Koguma, a micro-rover prototype for future lunar exploration missions designed and built at Tohoku University. This prototype is one of the smallest and lightest ever created weighing only 1 kg.

This paper presents an approach towards multi-rover coverage for the purpose of lunar surface resource prospecting. Considering the hardware limitations of current planetary exploration rovers, we focus on building a real time exploration algorithm for lunar surface coverage. Continuing with the aspect of limited rover capabilities, we also take into account the range limitations of the sensors by choosing a myopic approach. The rovers can only perceive the environment within a short range of their current position and decide their next actions depending on this information. These properties, along with the basic building blocks of the algorithm are explained in section 2. In section 3 we compare coverage results, first in the case of single rover autonomy, then for multi-rover systems. Finally, section 4 comments on the presented results and proposes the next steps to increase efficiency of the algorithms. Although this paper does not present results from field testing, the hardware limitations and capabilities of the Koguma platform were taken into consideration during development of the algorithms.

1.2 Related Work

The development of path planning strategies for planetary exploration comes with a most unique combination of constraints and difficulties to overcome. The soft soil covering all of the planetary surface, associated with a large number of unpredictable natural obstacle, make it difficult to achieve accurate planning over large

distances. A mission applicable exploration algorithm will have to take into account environmental conditions, limited sensor capabilities and be capable of on-the-fly replanning. In most current research for planetary exploration [14], the path planning algorithm consists of a graph-based method that minimises motion cost between a predefined start and goal position. The cost function used to give weight to the edges of the graph is defined as a combination of actual motion capabilities, terramechanical considerations and power consumption. The terramechanical factors are obtained experimentally and represent the capability of the rover to move in the lunar regolith. This is a good example of smart cost allocation based on environmental factors and rover mobility. The minimal cost path is finally obtained by applying Dijkstra's algorithm to the weighted map. In [12], the algorithm uses digital elevation maps and active visual based terrain classification to understand the surrounding environment (within a limited range) and generate its cost function. A final example in this line of exploration strategies is given in [13], where the variant this time is that the environment is perceived using a LIDAR, then segmented into a triangular mesh pattern which defines the graph and finally the path is obtained with A*. The main conclusions from this is that a usual algorithms should have to deal with limited sensing range and adaptability to both unplanned objects and environmental complexity.

Coverage Path Planning (CPP) includes a wide range of algorithms which aim to explore an environment in its entirety. Basic knowledge of this subject can be found in [1, 4, 7], where the authors have conducted surveys on a wide spread of methods and applications. An important aspect in CPP is environmental decomposition. Depending on the goal of the exploration method, the environment must be segmented into regions of various sizes and shapes. A path generation method is then used to explore all of these regions. One main aspect that makes these surveys notable is a listing of current coverage methods with their intended applications, advantages and shortcomings.

A popular approach to CPP is a graph based algorithm known as Spanning Tree Coverage (STC). In essence, STC computes a tree, spanning from a starting position for the rover and that branches out to every cell in the environment. Following this, each cell is subdivided into smaller sub-cells and the rover will circumnavigate the tree until it returns to its starting position. The three basic variants of STC (off-line, on-line and ant-like) are well explained in [2]. There are also variants of STC, called Spiral-STC and Scan-STC given in [3]. These two algorithms differ from the basic STC by the basic rover motion scheme where one uses a spiral like motion and the other a Boustrophedon pattern. They both also introduce a new system that allows for cell revisitation in order to properly explore the entirety of the exploration region. In [5], STC is applied to multi-rover CPP using a two step algorithm. The first steps generates a single STC solution that covers the entirety of the grid. The solution is then divided among the different rovers who execute the exploration task together. Finally, a different approach to multi-rover coverage is given in [6]. This algorithm takes into account a known environment and rover initial positions to divide the exploration region equally among the rovers then applies STC individually for each member. To summarise, graph based approaches and, in particular, STC represent

one branch of CPP but present some issues. These graph based approaches require prior knowledge of the environment and in particular, the location of obstacles. We present a path planning solution for environment coverage that does not require these high-resolution and accurate maps but instead, relies only on limited onboard sensor information.

For this paper, we need to define a simpler approach to coverage that supports unknown environments and can be expanded to multiple rovers operating in the same space. One such example is given in [11], where they use basic motion schemes and compare their efficiency. The objective is to map the edges of the environment and contours of obstacles. Although this is quite different from planetary exploration, the results obtained from each basic motion can be used and adapted to our needs. Another interesting algorithm for coverage in unknown environments is presented in [8]. This work does not focus on the motion scheme itself but instead, applies a simple Boustrophedon pattern and proposes an efficient backtracking method. Most CPP algorithms try to limit, if not suppress completely, backtracking as it is contrary to their definition of efficiency (i.e. shortest path and single cell visitation). The solution presented uses overall map geometry and memory of previously sensed, but unexplored cells to minimise total path length and ensure complete coverage. Although we also use similar basic motion patterns and allow for backtracking, we present methods that do not store or use past sensing information. Instead we compute motion at east step based on current sensor data and the exploration path up to that moment. Finally, an interesting concept towards solving the issue of multiple rovers in the same workspace is explained in [10]. Although it is fairly older than other related works introduced here, the algorithm utilizes myopic sensing, predefined decision based path planning and solutions for multi-rover coordination in unknown environments without communication between the rovers. What is most important about this work is the strict assumptions of fully decentralised computing for rovers with limited capabilities as these are also the foundations of our work.

2 Planetary Coverage for Multi-Rovers with Myopic Sensing

This section describes the theoretical aspects of a minimal CPP algorithm focussed on real-time application in unknown environments. Starting with the initial assumptions on the environment representation and rover capabilities (motion and sensing), we then move on to the single rover coverage algorithms. Finally, we present several scenarios to go from single to multi-rovers operating in the same environment. These algorithms focus on the idea of minimal computation usage for each step and, as such, the building blocks are in essence simple.

2.1 Initial assumptions

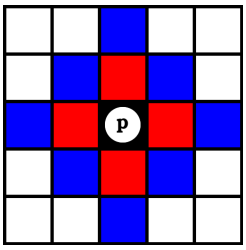
In this work the environment is represented as a 2D finite set of grid cells. Although that does seem like an unreasonable assumption for planetary exploration purposes, it serves first and foremost to develop and analyse a minimally complex algorithm. The lunar surface is virtually infinite on the scale of micro-rovers, but the finite aspect of the environment can be explained either by limitations set in the scope of the mission or more realistically, by the maximum communication range between the rovers and the lander. When deployed for an actual mission, it is fair to assume that some pre-mission knowledge of the terrain will be available (data from the Lunar Reconnaissance Orbiter or its successor) and that the rovers will be capable of detecting and understanding the features of the environment using their sensors. This data can then be compiled into a 2D height map and used to adapt the cost function associated with motion decisions. The 3D nature of the lunar surface can then be properly incorporated into the environment representation and enable the use of the hereafter presented algorithms in the context of planetary exploration.

Concerning the rovers, they are aware of their initial position and orientation. As they move in the environment, we consider them capable of SLAM, which means that they can accurately track their motion and obtain information about the surrounding cells. As one important factor towards minimal complexity, the sensor information is considered myopic. As illustrated in Fig.3a, the sensor information is available omni-directionally and within a limited range. Depending on the sensor array and its field of view, the shape and range of myopic sensing might differ from what is used here. This only represents a minor change in the algorithm and the results obtained will still be relevant. Using the sensor information, the rover then assigns static cost to motion depending on the nature of adjacent cells and its current orientation. The myopic sensing and cost allocation used for this paper are shown in Fig.3b. Although in reality, the time and power requirements are the same for both left and right turns, we decided to assign a different cost to these two. This takes away any random factor in the decision process by creating a preference of one over the other.

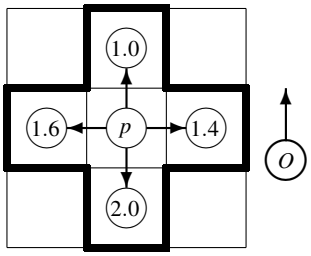
For multi-rovers, we consider they are homogeneous and all have the same basic capabilities presented above. If inter-rover communication is necessary, they can freely exchange information, are constantly within range and have no bandwidth limitations.

2.2 Single Rover Coverage

This part describes the building blocks used in order to achieve single rover coverage. The algorithms are based on two basic motion patterns onto which we add a few properties. The base exploration algorithm is detailed in 1. The different algorithm configurations are then tested and the results compiled in the following section.



(a) Two versions of myopic sensing with visibility at one and two steps ahead.



(b) Rover motion capabilities and cost with one step myopic sensing. The rover position if defined as p and current orientation by O.

Fig. 3: Illustration of omnidirectional myopic sensing (left) and motion cost attribution (right).

Algorithm 1 Single rover path planning
Input: Environment size (X,Y) and Percentage of coverage *CovPercent*
Output: Travelled path: *Path*

```
1: Initialise environment map (X,Y): All cell types set as unknown.
2: n= total number of cells
3: Set rover initial position and orientation
4: while the exploration is not finished do
5:     Get Sensor input
6:     Process input: Get neighbour relative coordinates and cell type from the map, add obstacles
   to map
7:     Compute motion cost for neighbours
8:     Move to lowest cost neighbour
9:     Add new position to Path and update the map
10:    for all cells in map do
11:        Count number of each cell type: V visited, O obstacle
12:    end for
13:    if V ≤ CovPercent ∗ (n − O) then
14:        Stop exploration
         return Path
15:    end if
16: end while
```

Basic motion: Boustrophedon, Fig.4a, and spiral motions, Fig.4b, are the two most basic motion options that exist. In very specific conditions, they are by themselves capable of achieving complete coverage. In most cases, and more specifically in an environment populated by obstacles, it is impossible to achieve single pass coverage and further properties are required to complete exploration of the environment.

Backtracking/cell revisitation: Since single pass complete coverage is impossible with the basic motion patterns, allowing cell revisitation is necessary. When no free cells are available within its neighbourhood, the rover is allowed to revisit cells.

(a) Boustrophedon motion. (b) Spiral motion.

Fig. 4: Representation of the two basic motion patterns used as the foundation of our work on a minimally complex coverage path planning algorithm.

As the rover explores the environment, it updates its internal map to keep track of cell revisitation.

Internal memory: With the hardware limitations and requirements of other systems, reducing memory usage in the path planning algorithm is also important. For this version of the algorithms, the rover stores the path it has travelled in a map of the environment and can access this information. This map is represented as a matrix of the same size as the exploration region. Cells are initialised as unknown and updated either as obstacles, if they enter the sensor field of view, or visited once the rover has passed over it. The map is used to determine if exploration is complete and to compute cost of neighbouring cells. With backtracking implemented, cells have an additional value stored in the map, V_i, representing the number of times that specific cell has been visited. Unknown and previously unvisited cells have a $V_i = 0$.

Motion cost function: The motion cost function is defined in Eq.1. For the neighbouring cell at coordinate (x,y), C_{Static} is a predefined value based on the position of the cell relative to the current orientation of the rover. If a cell has been previously visited, an additional cost is implemented to try and reduce as best as possible cell revisitation.

$$c(x,y) = C_{Static} + C_{Visited} V_i \tag{1}$$

Obstacle circumnavigation: In an attempt to improve efficiency, we wanted to include a specific property to handle obstacles. Similar to human behaviour, when a small obstacle appears in our path, the most logical approach is to simply go around it and continue on. In a similar fashion, obstacle circumnavigation can potentially be useful to improve coverage efficiency in terms of total path length.

Coverage percentage: Coverage completeness is defined by the fact that all cells, that are not obstacles, are visited at least once. In a realistic planetary exploration scenario, some cells might not be explorable, for example if they are surrounded by obstacles. Due to the minimal complexity of the algorithms presented here (no active use of memory), complete coverage would not be possible in this case. Assuming that these areas do not represent large portions of the exploration region, it would then be possible to achieve coverage up to predefined percentage.

2.3 Multi-Rover Coverage

After defining a minimal CPP algorithm for single rovers, we can now look into ways of having multiple rovers collaborate. Each rover uses an algorithm combining properties defined in the previous section. Two cases have been devised and will be compared through simulations.

Path overlapping: This case opens up the entirety of the environment to all rovers at the same time. The memory of cell visitation and additional cost of backtracking is common and shared among all members. For example, a given cell is visited by a first rover and marked in its path. At each step, rovers transmit their current position to all other members and their internal map is updated with this information. This will impact the cost of motion as defined in the single rover case. Exploration is considered finished when the map has been visited up to the predefined coverage percentage. The total path length is the same for each rover.

Environment sharing: Adding an initial step before the CPP algorithm is started, this method divides the exploration region among rovers. Based on Voronoï partitioning, the environment is divided among each member depending on their initial position. The rovers then have their individual exploration regions and apply a single rover CPP algorithm within it. In this case, each rover will stop once they achieve coverage, up to the predefined percentage, in their assigned region. The path length used in the evaluation of simulation results is the total path length for the last rover to complete exploration, the longest travelled path.

3 Simulations and Results

In this section, we present simulation results for the algorithms introduced in the previous section. First, we describe the simulation environment characteristics and properties. Then, we evaluate the efficiency of single rover exploration methods. Finally, using one of these algorithms, we compare the simulation results of multi-rover exploration methods.

3.1 Environment Set-Up

As described in Section 2.1, the test environment is represented as a 2D grid composed of free cells and obstacles. For all simulations, we can control the size of the grid and percentage of obstacles. The location of these obstacles is randomly assigned, but maps can be saved and reused in later simulations. The starting position and orientation of each rover can either be assigned or randomly selected. This allows us to run a variety of simulations, testing the influence of several parameters on efficiency.

Like most other coverage algorithms, one aspect of efficiency is obtaining the shortest travelling path. To visualise and compare the results of different simulations, we introduce the path length ratio, defined in equation 2. For single rover exploration, a value of 1.0 means the algorithm is capable of visiting the entirety of the exploration region with no backtracking and passing over all cells exactly once.

$$Path\ length\ ratio = \frac{Total\ path\ length}{Total\ number\ of\ cells} \tag{2}$$

After conducting a large number of simulations and using this value, we evaluate the individual efficiency of each algorithm and compare the results between different methods.

3.2 Single Rover Coverage Results

Using the properties defined in section 2.2, we first compare four basic algorithms: Spiral and Boustrophedon motion, both with and without obstacle circumnavigation. The typical flow of a simulation run is illustrated in Fig.5. Between the last two steps shown here, only a few cells remain however the path length ratio goes from 1.2 to 1.75. By reducing the coverage percentage to 95% (achieved by the third image), most of the environment would still be explored but it would represent a substantial decrease in total path length and thus, exploration time.

Random assignment of obstacle location can sometimes lead to small closed off regions that are inaccessible to the rover. Although this is more realistic in the context of planetary exploration is does prevent these simple algorithms from achieving complete coverage. In order to properly compare the effect of coverage percentage on the total path length of the single rover methods, we assume in this section only that all free cells in the environment are accessible. The results are presented in Fig.6, and shows the efficiency of each single rover algorithm depending on a required coverage percentage.

Although all versions of the single rover coverage algorithm have the same degree of complexity from a computational aspect, the Boustrophedon motion scheme shows severe inefficiency compared to a spiral approach when considering total travelled distance. When looking at the spiral approach, we can observe a lower average motion for all percentages of coverage completion when not applying obstacle circumnavigation. Adding this behaviour has not induced an increase in computational requirements, but the higher average in path length make its usage unnecessary and means it can be removed from the exploration algorithm. Considering completeness, we still want to visit as much of the area as possible while reducing overall exploration time. When the algorithm is close to completion, only a few scattered cell remain around the map. The simulation results show a 10% to 20% increase in path length just to accomplish the last 2% of exploration. The standard deviation for the complete algorithm is twice the value obtained for 97% in the case of methods based on spiral motion. As a sufficient compromise between maximum exploration

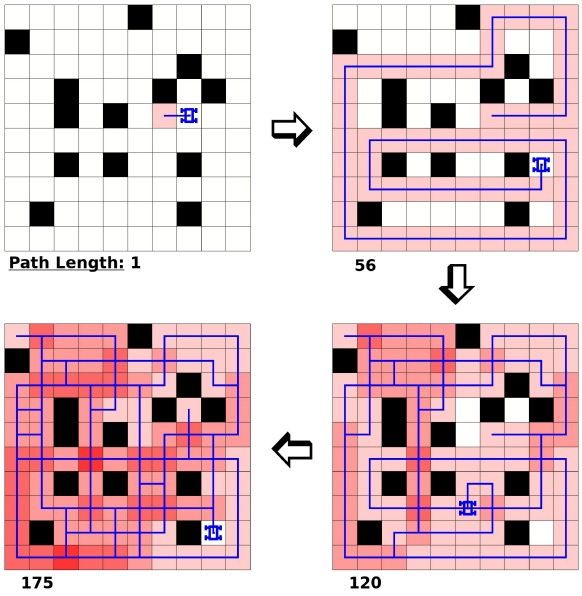


Fig. 5: Representation of a typical simulation run from initialisation (top left) to exploration completion (bottom left). The intensity of colour for each cell indicates the number of times the cell has been visited. Darker shades mean the cell has been visited more often. The number under each image indicates the current path length at that stage of exploration.

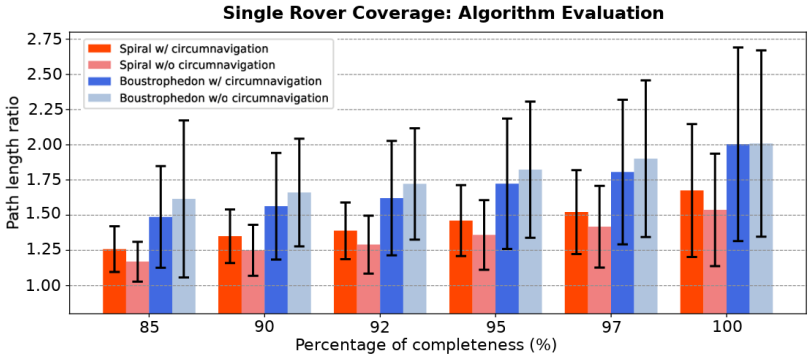
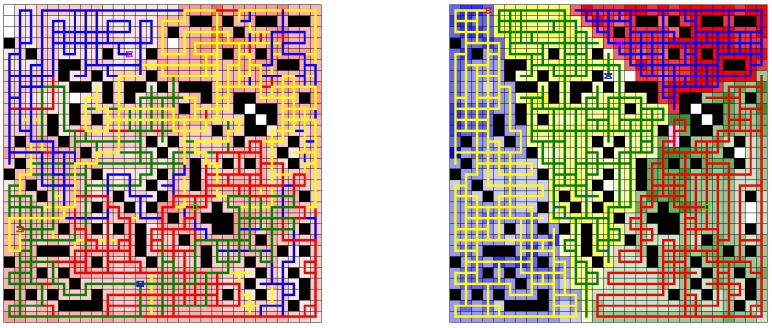


Fig. 6: Comparison of efficiency using single rover algorithms for various percentages of completeness. Results are based on a larger number of simulations (3000 runs per exploration algorithm and for each percentage of completeness) in multiple environments with random starting positions every time.

and efficiency, we choose 95% completeness as the end goal for the following simulations.

3.3 Multi-Rover Coverage Results

In Section 2.3, we introduced two potential methods to achieve multi-rover coverage. Using several numbers of rovers in environments of different sizes, we conducted simulations with the goal of achieving a predefined coverage percentage. Using four rovers and for a required percentage of 99%, we obtained the result shown in Fig.7. Although their appearance is vastly different, the final result is almost the same with approximately the same path length ratio. This is the main reason why these two results were selected as illustrations here but this is not a guaranteed result as we will now show.



(a) Algorithm with overlapping paths. (b) Algorithm with Voronoï decomposition.

Fig. 7: Representation of the simulations obtained for both multi-rover methods presented in this paper. The overlapping method achieves coverage in 444 steps and the Voronoi based method in 449 steps. The required coverage percentage in these test was set to 99%.

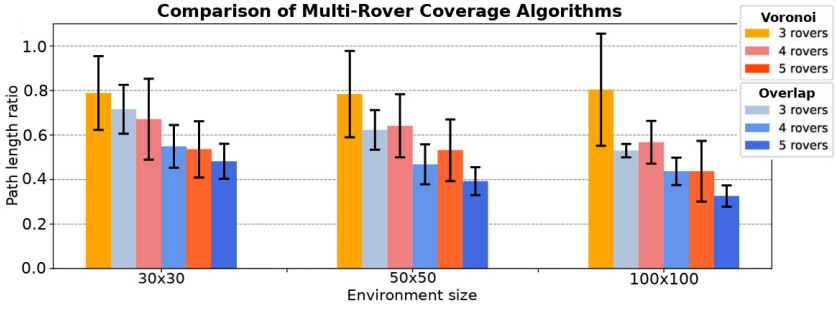


Fig. 8: Results of multi-rover coverage using overlapping exploration regions and Voronoï partitioning.

The simulation results of both algorithms are presented in Fig.8. The testing conditions include several grid sizes and number of rovers used for exploration. The required coverage percentage for these tests was set to 95%. The results show that the method allowing path overlapping for the rovers systematically outperforms separate exploration with Voronoï partitions. The partitioning method is highly dependant on the initial positions and is only optimal in the rare circumstance that the rovers are perfectly placed for even distribution. This is shown by the much higher path length and large standard deviation. One solution to ensure optimality would be to initially reposition the rovers before segmenting the map. On the other hand, the overlapping method is truly a collaborative exploration method where the conjunct efforts of rovers allow for faster convergence to unexplored areas and shorter total path length. In addition to this, the overlapping method does not require any computation before starting exploration.

In conclusion, the overlapping based method allows faster coverage with less computational requirements. The small standard deviation, in all configurations presented here, guarantee consistency of the results. With some small upgrades and optimisation techniques, this approach could represent the first step in developing a functional multi-rover path planning solution for collaborative planetary exploration.

4 Conclusions and Future Developments

In this paper we have introduced and analysed the efficiency of some basic single and multi-rover exploration strategies. Focussing first and foremost on simple algorithms for real-time applications, we have performed simulations to evaluate their potential to achieve coverage. For the single rover algorithm, complete coverage can only be achieved if all free cells of the environment are accessible. In a more realistic scenario, some smaller regions would be blocked by the presence of obstacles. To overcome this, additional properties based on the use of internal memory and optimisation techniques will need to be implemented. Additionally, we discovered that completing the final few percent of exploration severely impacts the performance of the algorithm and it is better to reduce slightly the percentage of coverage. For multi-rover methods, we compared the results of two algorithms, one based on Voronoï partitioning the environment among the rovers and the other allowing for full overlapping of the group's exploration region. In the case of segmented environments, the increased computation in the initialisation step can only be justifiable if we can equally divide the area among the multiple rovers to obtain equal path lengths for each rover. With unknown environments and more so random starting positions, the segmentation based method is more often less efficient than a free roaming group. Although it is mostly compared to worst case versions of the Voronoï method, it did show on average better and more reliable results. It is a true collaborative exploration method and will be used as the foundation for our next steps in development.

Using the results obtained from this study, we first plan on developing and integrating some aspects of optimisation. Sticking with the idea that real-time path generation is a must for future mission implementations, these more capable algorithms will be benchmarked for onboard computation using the Koguma platform, Fig.2. In parallel, we will conduct field tests for the multi-rover systems to assess viability of each advances in algorithm, identify key issues and speed up development of software for real mission applications.

References

1. Choset, H.: Coverage for robotics–A survey of recent results. *Annals of mathematics and artificial intelligence* **31(1-4)**, 113–126 (2001)
2. Gabriely, Y., Rimon, E.: Spanning-tree based coverage of continuous areas by a mobile robot. *Annals of mathematics and artificial intelligence* **31(1-4)**, 77–98 (2001)
3. Gabriely, Y., Rimon, E.: Competitive on-line coverage of grid environments by a mobile robot. *Computational Geometry* **24(3)**, 197–224 (2003)
4. Galceran, E., Carreras, M.: A survey on coverage path planning for robotics. *Robotics and Autonomous systems* **61(12)**, 1258–1276 (2013)
5. Hazon, N., Kaminka, G.A.: Redundancy, efficiency and robustness in multi-robot coverage. *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 735–741 (2005)
6. Kapoutsis, A.C., Chatzichristofis, S.A., Kosmatopoulos, E.B.: DARP: divide areas algorithm for optimal multi-robot coverage path plannings. *Journal of Intelligent & Robotic Systems* **86(3-4)**, 663–680 (2017)
7. Khan, A., Noreen, I., Habib, Z.: On Complete Coverage Path Planning Algorithms for Non-holonomic Mobile Robots: Survey and Challenges. *Journal of Information Science and Engineering* **33(1)**, 1258–1276 (2017)
8. Khan, A., Noreen, I., Ryu, H., Doh, N.L. Habib, Z.: Online complete coverage path planning using two-way proximity search. *Intelligent Service Robotics* **10(3)**, 229–240 (2017)
9. Laîné, M., Tamakoshi, C., Touboulic, M., Walker, J., Yoshida, K.: Initial Design Characteristics, Testing and Performance Optimisation for a Lunar Exploration Micro-Rover Prototype. *Advances in Astronautics Science and Technology* **1(1)**, 111–117 (2018)
10. Lumelsky, V.J., Harinarayan, K.R.: Decentralized motion planning for multiple mobile robots: The cocktail party model. *Autonomous Robots* **4(1)**, 121–135 (1997)
11. Masehian, E., Jannati, M. Hekmatfar, T.: Cooperative mapping of unknown environments by multiple heterogeneous mobile robots with limited sensing. *Robotics and Autonomous Systems* **87**, 188–218 (2017)
12. Ono, M., Fuchs, T.J., Steffy, A., Maimone, M., Yen, J.: Risk-aware planetary rover operation: Autonomous terrain classification and path planning. *IEEE Aerospace Conference*, 1–10 (2015)
13. Rekleitis, I., Bedwani, J.L., Dupuis, E.: Autonomous planetary exploration using LIDAR data. *IEEE International Conference on Robotics and Automation*, 3025–3030 (2009)
14. Sutoh, M., Otsuki, M., Wakabayashi, S., Hoshino, T., Hashimoto, T.: The right path: comprehensive path planning for lunar exploration rovers. *IEEE Robotics & Automation Magazine* **22(1)**, 22–33 (2015)
15. Tuna, G., Gulez, K., Gungor, C.V., Mumcu, V.T.: Evaluations of different simultaneous localization and mapping (SLAM) algorithms. *IECON 2012-38th Annual Conference on IEEE Industrial Electronics Society* 2693–2698 (2012)
16. Walker J .: Final configuration of the ispace HAKUTO rover for a Google Lunar XPRIZE mission. *68th International Astronautical Congress* (2017)