

## RAPF

### Efficient path planning for lunar microrovers

Manteaux, Thomas; Rodriguez-Martinez, David; Rajan, Raj Thilak

#### DOI

[10.1109/iSpaRo60631.2024.10688117](https://doi.org/10.1109/iSpaRo60631.2024.10688117)

#### Publication date

2024

#### Document Version

Final published version

#### Published in

2024 International Conference on Space Robotics, iSpaRo 2024

#### Citation (APA)

Manteaux, T., Rodriguez-Martinez, D., & Rajan, R. T. (2024). RAPF: Efficient path planning for lunar microrovers. In *2024 International Conference on Space Robotics, iSpaRo 2024* (pp. 56-63). (2024 International Conference on Space Robotics, iSpaRo 2024). IEEE.  
<https://doi.org/10.1109/iSpaRo60631.2024.10688117>

#### Important note

To cite this publication, please use the final published version (if applicable).  
Please check the document version above.

#### Copyright

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

#### Takedown policy

Please contact us and provide details if you believe this document breaches copyrights.  
We will remove access to the work immediately and investigate your claim.

***Green Open Access added to TU Delft Institutional Repository***

***'You share, we take care!' - Taverne project***

***<https://www.openaccess.nl/en/you-share-we-take-care>***

Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public.

# RAPF: Efficient path planning for lunar microrovers

Thomas Manteaux<sup>1,2</sup>, David Rodríguez-Martínez<sup>2</sup>, and Raj Thilak Rajan<sup>1</sup>

<sup>1</sup>Signal Processing Systems, Faculty of EEMCS, Delft University of Technology, The Netherlands

<sup>2</sup>Advanced Quantum Architecture Lab (AQUA), Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland

**Abstract**—Efficient path planning is key for safe autonomous navigation over complex and unknown terrains. Lunar Zebro (LZ), a project of the Delft University of Technology, aims to deploy a compact rover, no larger than an A4 sheet of paper and weighing not more than 3 kilograms. In this work, we introduce a Robust Artificial Potential Field (RAPF) algorithm, a new path-planning algorithm for reliable local navigation solution for lunar microrovers. RAPF leverages and improves state of the art Artificial Potential Field (APF)-based methods by incorporating the position of the robot in the generation of bacteria points and considering local minima as regions to avoid. We perform both simulations and on field experiments to validate the performance of RAPF, which outperforms state-of-the-art APF-based algorithms by over 15% in reachability within a similar or shorter planning time. The improvements resulted in a 200% higher success rate and 50% lower computing time compared to the conventional APF algorithm. Near-optimal paths are computed in real-time with limited available processing power. The bacterial approach of the RAPF algorithm proves faster to execute and smaller to store than path planning algorithms used in existing planetary rovers, showcasing its potential for reliable lunar exploration with computationally constrained and energy constrained robotic systems.

**Index Terms**—Path planning, Artificial Potential Field, lunar rover, microrovers

## I. INTRODUCTION

Despite significant advancements in space technology and robotics, navigating efficiently across unknown and unstructured environments remains an ongoing pursuit. This paper explores the importance of efficient path planning for small lunar rovers (<100 kg) and microrovers (<10 kg), which have gained significance with the advent of cheaper and commercial exploration missions over the last decade. Several state-of-the-art lunar rover path planning solutions have been developed, drawing from diverse fields such as robotics, artificial intelligence, and space exploration. For instance, Yu et al. [1] propose a heuristic-based approach that considers terrain complexity and energy efficiency for global path planning on the lunar surface. Another notable work by Peng et al. [2] utilizes deep neural networks to perform data-driven path planning on the basis of heuristic search. These advancements showcase the ongoing efforts to enhance the autonomy and efficiency of lunar exploration missions through innovative

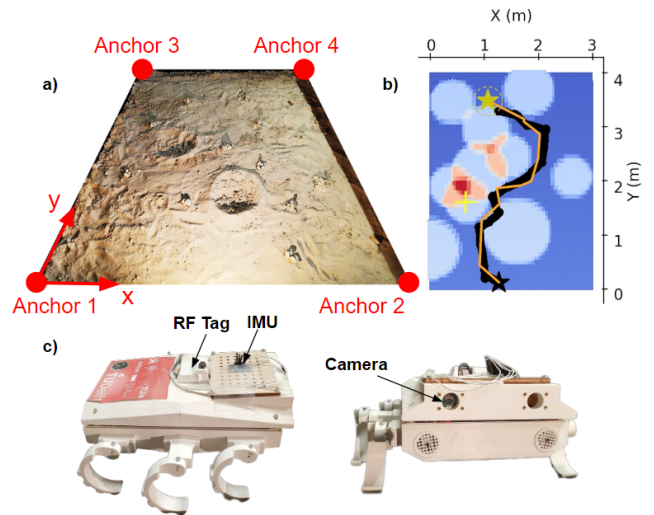


Fig. 1. a) Field test layout. b) Potential map computed by the RAPF algorithm. Dark blue represents low-potential regions while red represents high-potential regions. The orange line shows the computed path, black line defines the ground truth, and the yellow cross shows where a local minimum was found. c) Lunar Zebro prototype used for testing

path planning techniques. Although recent trends, such as deep learning, show promise, they remain computationally intensive, as compared to more resource efficient algorithms e.g., Rule-based approaches, including Dijkstra-based [3], Artificial Potential Field (APF)-based [13], and sampling-based [5] algorithms. Small lunar rovers, operating in complex environments with communication delays, restricted computational, memory and power capabilities, and limited sensor suites demand robust, real-time path planning algorithms. These algorithms must cope with the irregularities of the lunar surface—characterized by uneven and unconsolidated terrains filled with craters and rocks of different sizes—while addressing measurement noise from sensors on-board the rover. Due to their small size and reduced mobility, microrovers face challenges in negotiating rough or uneven terrain. Small size constrains sensor range, limiting the perception of surrounding environments for micro-rovers preventing comprehensive perception and posing challenges for effective path planning in navigation. The power constraint of lunar rovers demands

low-complexity and efficient path planning, striking a balance between computational cost and performance. Path planning emerges, therefore, as a critical aspect impacting mission efficiency and success.

#### A. Contributions

To robustly and efficiently compute the path to be followed by a lunar microrover, we propose a new lightweight solution, RAPF, based on adapting and expanding an existing APF-based algorithm. Our approach is adapted to run onboard the LZ walking rover on a Raspberry Pi 4B. We demonstrate the performance of our algorithm both in simulations and during field tests across a lunar-like environment. Results show the greater accuracy and lower memory allocation requirements of our approach, displaying comparable performances even when compared to optimal terrestrial algorithms such as A\*.

#### B. Related work

Different approaches exist for computing the optimal path to follow between points of interest.

*Combinatorial methods*—which include the Dijkstra, A\* [3], and D\* [4] algorithms—explore discrete paths through a graph representation of the environment. Graph nodes represent specific configurations of the robot. Edges represent feasible transitions between configurations. Only transitions between neighboring graph nodes are allowed. Graph search algorithms explore the graph by visiting nodes and searching for the optimal path based on certain criteria such as distance or cost. Despite ensuring optimal paths and completeness, combinatorial methods demand significant memory and computational capabilities, struggling in environments where the map is frequently recomputed.

*Sampling-based algorithms* divide a continuous space into sample points forming a roadmap or tree, with Rapidly Exploring Random Tree (RRT) [5] and Probabilistic Roadmaps (PRM) [6] as prominent examples of this approach. Sampling involves a deterministic or stochastic selection of points within the configuration space, representing potential robot configurations. These sampled points stand as potential configurations that the robot can take. The objective is to comprehensively cover the configuration space to unveil feasible paths. The connection of these points forms a graph structure, balancing optimality and completeness. However, these algorithms can be sensitive to the initial configuration [7] and may necessitate time-intensive pre-processing.

Biologically inspired methods such as *spiking neural models* [8] use artificial neural networks to mimic the brain's structure, enabling robots to adaptively plan paths based on sensory inputs. Swarm intelligence, inspired by collective behavior, uses algorithms like Particle Swarm Optimization [9] and Ant Colony Optimization [10]. Genetic algorithms [11], derived from natural selection principles [12], iteratively refine potential solutions. As they replicate problem-solving strategies found in nature, biologically inspired methods offer a large adaptability making them well-suited in complex and dynamic environments. They provide robustness against unpredicted obstacles and changing surroundings. However, they require a large number of parameters to be fine-tuned and

their implementation is usually complex. They therefore have a large computational cost. Biologically inspired methods do not always outperform traditional path planning methods – combinatorial, sampling-based and APF-based methods – in static environments and their complexity might hinder their implementation on embedded systems working in real-time.

Lastly, potential field techniques [13] guide robots using artificial forces. In the *Artificial Potential Field* (APF) method, an attractive potential pulls the robot towards the target, while repulsive potentials keep it away from obstacles. The APF approach ensures collision-free paths in real-time with low computational complexity. However, they face local minima issues, potentially trapping the robot in sub-optimal positions. Local minimas occur due to various reasons such as symmetry between the different obstacles, closely spaced obstacles, or in highly cluttered environments. Tuning the balance between repulsive and attractive potentials is crucial for successful path planning and to avoid local minima. The next section presents the original APF algorithm and some of its more promising variations. These form the basis of the approach proposed herein.

## II. ARTIFICIAL POTENTIAL FIELD (APF) ALGORITHM

The classical APF was originally proposed in [13], in which a robot navigates in an artificial force field, with attractive forces being emitted by the target and repulsive forces by the obstacles. These forces are directly derived from quadratic potential functions, which are parameterized by the robots distance to a target or an obstacle. The robot, therefore, aims at minimizing its potential cost function at any given point along its' path. A local minima event occurs, when no adjacent point exists in the neighborhood of the rover with a lower potential as compared to the current position of the robot. The attractive potential acting on a robot at the global position  $\mathbf{x} = [x, y]^T$  in a 2-dimensional scenario, from a target located at  $\mathbf{x}_t = [x_t, y_t]^T$  is defined as

$$J_a(\mathbf{x}) = \frac{1}{2}k_a d_t^2(\mathbf{x}), \quad (1)$$

where  $k_a$  is the attractive gain,  $d_t(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_t\|_2$  is the Euclidean distance between the robot and the target. The repulsive potential acting on a robot located at  $\mathbf{x}$  from the  $n$ th detected obstacle at location  $\mathbf{x}_o^n = [x_o^n, y_o^n]^T$  is defined as

$$J_o^n(\mathbf{x}) = \begin{cases} \frac{1}{2}k_r \left( \frac{1}{d_n(\mathbf{x})} - \frac{1}{\rho_0} \right)^2 & \text{if } d_n(\mathbf{x}) \leq \rho_0, \\ 0 & \text{otherwise,} \end{cases} \quad (2)$$

where  $k_r$  is the repulsive gain,  $\rho_0$  the maximum distance within which the potential field of the obstacle exerts its influence, and  $d_n(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_o^n\|_2$  is the Euclidean distance between the robot and the  $n$ th detected obstacle. Let  $N_{obs}$  be the number of obstacles detected by the robot at position  $\mathbf{x}$ , then the total repulsive potential acting on the robot is

$$J_r(\mathbf{x}) = \sum_{n=1}^{N_{obs}} J_o^n(\mathbf{x}) \quad (3)$$

and the total potential function is the sum of the attractive and the repulsive potentials as stated below

$$J(\mathbf{x}) = J_a(\mathbf{x}) + J_r(\mathbf{x}). \quad (4)$$

The robot's direction of motion is determined by calculating the gradient relative to the potential of its current position.

#### A. RVF

An alternative potential field method for escaping local minima is proposed in [14] and [15], called the Rotated Vector Field (RVF), which reorients the force field directly derived from the classical APF potential functions as described below. The resulting attractive force is defined as the gradient function of the attractive potential

$$\mathbf{F}_a(\mathbf{x}) = -\nabla J_a(\mathbf{x}), \quad (5)$$

where  $J_a(\mathbf{x})$  is defined in (1). Similar to the attractive force, the resulting repulsive force is directly derived from the potential function as

$$\mathbf{F}_r^n(\mathbf{x}) = -\nabla J_o^n(\mathbf{x}), \quad (6)$$

where  $J_o^n(\mathbf{x})$  is defined in (2). Similar to the total force field defined for APF in (4), the total force is the sum of the attractive and repulsive forces written as

$$\mathbf{F}(\mathbf{x}) = -\nabla J(\mathbf{x}) = -(\nabla J_a(\mathbf{x}) + \nabla J_r(\mathbf{x})). \quad (7)$$

The main benefit of this approach is that instead of being repelled when approaching an obstacle, the robot is forced to move around it. This is achieved by rotating the vector field around the obstacles in one direction or another. This creates a vortex field at the location of the  $n$ th detected obstacle given by

$$F_n(\mathbf{x}) = \pm \begin{pmatrix} \frac{\partial J_{t,n}(\mathbf{x})}{\partial y} \\ -\frac{\partial J_{t,n}(\mathbf{x})}{\partial x} \end{pmatrix}. \quad (8)$$

The intensity of the vector field remains the same, only the direction changes. The vortex field is bounded at a distance  $\rho_0$  from the obstacles, the maximum distance within which the potential field of the obstacle exerts its influence, such that the entire vector field does not take on this rotary shape.

#### B. CRBAPF\*

In [16] and [17] a set of enhanced APF algorithms called Bacteria APFs (BAPFs) are proposed. These algorithms involve what is called *bacteria points*, i.e., virtual points around the robot representing a potential solution to the optimization problem of the APF algorithm. A new APF algorithm based on the BAPF concept was introduced in [18], named Changing Radii Bacteria APF (CRBAPF\*), which actively modifies the potential functions and enables a random walk technique (RWT) to escape local minima. The potential functions are computed for the bacteria points by taking into account their distances from the target and any obstacles detected in the environment. The bacteria with the lowest cost is chosen as the next position for the robot, and the latter moves towards that selected point. The cost functions from the target and the obstacles are expressed as Gaussian functions instead of quadratic functions in the classical APF method. Using Gaussian over quadratic cost functions presents several advantages such as a more gradual increase in repulsion as the robot approaches an obstacle and a smoother transition from attractive to repulsive forces. This solution results in smoother

trajectories, a more natural behavior, and a better handling of narrow passages. The attractive potential acting on a robot located at  $\mathbf{x} = [x, y]^T$  from a target located at  $\mathbf{x}_t = [x_t, y_t]^T$  is defined as

$$J_a(\mathbf{x}) = -\alpha_a e^{-\mu_a d_t(\mathbf{x})}, \quad (9)$$

with  $\alpha_a$  and  $\mu_a$  constant parameters established through empirical means and they can be seen as the height and the width of the potential source.  $d_t(\mathbf{x})$  is the squared Euclidean distance between the robot and the target. The repulsive potential acting on a robot located at  $\mathbf{x}$  from the  $n$ th detected obstacle at location  $\mathbf{x}_o^n = [x_o^n, y_o^n]^T$  is defined as

$$J_o^n(\mathbf{x}) = \begin{cases} 0 & \text{if } \sqrt{d_n(\mathbf{x})} > \rho_u, \\ \alpha_o e^{-\mu_o d_n(\mathbf{x})} & \text{if } \rho_l < \sqrt{d_n(\mathbf{x})} < \rho_u, \\ \infty & \text{if } \sqrt{d_n(\mathbf{x})} < \rho_l, \end{cases} \quad (10)$$

with  $\rho_l$  the lower radius and  $\rho_u$  the upper radius setting the boundaries of the potential source.  $\alpha_o$  and  $\mu_o$  are constants tuned by testing and dependent on the average density of obstacles, and  $d_n(\mathbf{x})$  is the squared Euclidean distance between the robot and the  $n$ th detected obstacle. For a number  $N_{obs}$  of obstacles detected by the robot at a position  $\mathbf{x}$ , the total repulsive potential acting on the robot is

$$J_r(\mathbf{x}) = \sum_{n=1}^{N_{obs}} J_o^n(\mathbf{x}), \quad (11)$$

and the total potential function is the sum of the attractive and the repulsive potentials as stated below.

$$J_r(\mathbf{x}) = J_a(\mathbf{x}) + J_r(\mathbf{x}). \quad (12)$$

#### Bacteria potential

For a robot located at  $\mathbf{x} = [x, y]^T$ , the position of the  $j$ th the bacteria is  $\mathbf{x}_{b,j} = [x_{b,j}, y_{b,j}]^T$ , defined as

$$\begin{pmatrix} x_{b,j} \\ y_{b,j} \end{pmatrix} = \begin{cases} x + \rho_b \cos\left(\frac{2\pi n_j}{N_B}\right), \\ y + \rho_b \sin\left(\frac{2\pi n_j}{N_B}\right), \end{cases} \quad (13)$$

where  $N_B$  is the number of bacteria,  $n_j = 1, 2, \dots, N_B$  and  $\rho_b$  the radius around the robot where bacteria are generated. Similarly to the rover, the total potential function at the location of the  $j$ th bacteria is expressed as

$$J_{b,j}(\mathbf{x}_{b,j}) = J_a(\mathbf{x}_{b,j}) + J_r(\mathbf{x}_{b,j}). \quad (14)$$

As the robot is looking for the lowest potential towards the target, the next selected bacteria point  $j$  shall meet the criterion

$$J_{b,j}(\mathbf{x}_{b,j}) - J_r(\mathbf{x}) < 0. \quad (15)$$

If several bacteria points meet (15), the one with the lowest Euclidean distance toward the target is selected.

#### Random walk technique

To prevent the robot from getting stuck in a local minimum, a random walk technique is employed. The robot randomly selects a bacteria point around it, moves to it, and repeats the process for a pre-defined amount of time. The random selection of the bacteria point follows a uniform distribution. The only condition to be met follows from (10).

TABLE I  
OVERVIEW OF STATE-OF-THE-ART APF-BASED ALGORITHMS

Algorithm	APF	RVF	CRBAPF*	RAPF
<b>Key concept</b>	Attractive/Repulsive potential	Rotated vector field	Bacteria points + RWT	Bacteria points + artificial obstacles
<b>Cost function type</b>	Quadratic	Quadratic	Gaussian	Gaussian
<b>Handling Local minima</b>	-	Rotation of the field	Random walk technique	Artificial obstacles
<b>Trajectory smoothness</b>	Abrupt changes	Smooth trajectories	Oscillations for low $N_B$	Smooth trajectories

### C. Robust and Lightweight APF

Our proposed approach in this work, the Robust and Lightweight APF (RAPF) method, differs from the CRBAPF\* algorithm in two different ways. First, RAPF uses a different local minima escape approach instead of the random walk technique. In RAPF, when local minimas are encountered, then they are labeled as artificial obstacles. These artificial obstacles generate a repulsive potential (Eq. 10) that influences the total potential function (Eq. 12). The existing planned path is cleared and a new path is computed from the current position, which includes the local minima as an obstacle. Another improvement introduced into the RAPF algorithm consists of taking into account the position of the robot with respect to the target during the generation of the bacteria points. The  $N_B$  bacteria points are generated such that there is always a bacteria point along the robot-target vector. This removes the oscillation effect from the CRBAPF\* algorithm when a small number of bacteria points are used. This feature enables the RAPF algorithm to use a reduced number of bacteria points ( $N_B = 8$ ), resulting in a more lightweight method. Algorithm 1 describes the working principle of the RAPF algorithm.

At the onset crucial parameters are set-up like observed obstacles, the number of bacterial agents, movement step size, and maximum duration. It initializes key variables such as the current position, a timer for elapsed time tracking, and a boolean flag for path determination. Within each iteration, the algorithm assesses the robot's position by calculating the cost function (J) considering obstacle influence. It computes the cost function for each bacteria point ( $Idx_B$ ) and selects a new robot position based on these evaluations, appending it to the path. If a local minimum is detected, the algorithm resets the path to the starting position and updates observed obstacles. It also checks if the maximum time has been exceeded or if the robot is within the target margin. If any condition is met, the algorithm terminates and returns the path; otherwise, it iterates until finding a satisfactory path or reaching the time limit.

The key properties of the different APF algorithms including RAPF are listed in Table I.

## III. SIMULATIONS

### A. Virtual environment

To test the performance of our algorithms in a more realistic, we perform simulations in a virtual environment consisting of rocks and craters on Lunar surface. We achieve this by studying the size-frequency and cumulative number

### Algorithm 1 RAPF algorithm

**Require:**  $\mathbf{x}, \mathbf{x}_t, N_{obs}, N_B, StepSize, MaxTime, \{\mathbf{x}_i, r_i\}_{i \in N_{obs}}, GoalMargin$   
**Ensure:** *PathToTarget*  
1:  $N_{obs} = \{1, 2, \dots, N_{obs}\}$   
2:  $N_B = \{1, 2, \dots, N_B\}$   
3:  $Path = \mathbf{x}$   
4:  $Timer = CurrentTime$   
5:  $FindingPath = True$   
6: **while**  $FindingPath$  **do**  
7:  $J_{a,robot} \leftarrow get\_Ja(Path, \mathbf{x}_t)$   
8:  $J_{r,robot} \leftarrow get\_Jr(Path, \{\mathbf{x}_i, r_i\}_{i \in N_{obs}})$   
9:  $J_{robot} \leftarrow J_{a,robot} + J_{r,robot}$   
10: **for**  $Idx_b \in N_B$  **do**  
11:  $\mathbf{x}_b \leftarrow get\_bacteria(Path, StepSize, N_B, Idx_b)$   
12:  $J_{a,bacteria} \leftarrow get\_Ja(\mathbf{x}_b, \mathbf{x}_t)$   
13:  $J_{r,bacteria} \leftarrow get\_Jr(\mathbf{x}_b, \{\mathbf{x}_i, r_i\}_{i \in N_{obs}})$   
14:  $J_{bacteria} \leftarrow J_{a,bacteria} + J_{r,bacteria}$   
15: **end for**  
16:  $new_x \leftarrow pick\_bacteria(Path, J_{robot}, J_{bacteria}, N_{obs})$   
17:  $Path \leftarrow Add(new_x, Path)$   
18: **if**  $local\_minimum(Path)$  **then**  
19:  $Path = \mathbf{x}$   
20:  $N_{obs} \leftarrow Add(\mathbf{x}_{loc\_min}, \{\mathbf{x}_i, r_i\}_{i \in N_{obs}})$   
21: **end if**  
22: **if**  $Timer - CurrentTime > MaxTime$  **then**  
23:  $FindingPath = False$   
24: **end if**  
25: **if**  $\|\mathbf{x}_t - \mathbf{x}_j\|_2 < GoalMargin$  **then**  
26: **return**  $Path$   
27: **end if**  
28: **end while**  
29: **return**  $None$

distributions of these geographical features, which are derived from past lunar mission data. Based on data from [19], the cumulative fractional area covered by rocks with a diameter greater than D versus rock diameter (D) is given by

$$F_r(D) = k_r e^{-q_r(k_r) * D}, \quad (16)$$

where  $k_r$  is the rock abundance [%] and  $q_r [m^{-1}]$  is a coefficient determined empirically. From (16) the cumulative number of rocks per square meter versus rock diameter ( $D_r$ ) is then given by

$$N_r(D) = \frac{4q_r(k_r)k_r}{\pi} \left( \frac{e^{-q_r(k_r)D}}{D} - q_r(k_r) Ei(-q_r(k_r)D) \right). \quad (17)$$

The rock abundance is expected to be smaller than 10% as observed in previous NASA and ESA missions [20], [21]. Extending values from previous missions (Apollo, Chang'e, and Luna) to the LZ landing site we obtain  $k_r$

$= 0.02$  and  $q_r = 1.6 \text{ m}^{-1}$ . The height-to-diameter ratio allows us to determine that rocks with a diameter greater than  $D_{r,crit} = 6.5\text{cm}$  are considered obstacles for LZ, i.e. a height greater than 35 mm. The crater abundance model uses the same (16) and (17). A Monte-Carlo simulation is run to create three different scenarios in which to compare the previously introduced algorithms. The map is  $30 \times 30 \text{m}^2$  with a distribution of the obstacles within  $(x_{min}, y_{min}) = (5, 5)$  and  $(x_{max}, y_{max}) = (25, 25)$ . The rocks cover 1.8% and the craters 15% of this  $20 \times 20 \text{m}^2$  area. The starting position is located at  $(x_s, y_s) = (2, 2)$  and the target position is a circle centered at  $(x_g, y_g) = (28, 28)$ , with a radius of 0.5m. The radius of the rover is 0.20m. A\* is used as the optimal reference for this analysis. Table II presents the distribution of rocks and craters for each scenario. Despite the large differences between the numbers, the obstacle coverage of the surface is constant as previously defined ( $A_{rock} = 1.8\%$ ,  $A_{crater} = 11\%$ ). 500 simulations are run on each environment for each algorithm. The position of obstacles is randomly generated for each new simulation.

TABLE II  
NUMBER OF OBSTACLES FOR EACH SCENARIO

Scenario	Rocks	Craters	Total obstacles
A	42	38	80
B	88	32	120
C	137	24	161

### B. Performance metrics

The following metrics are used to evaluate the performance of the algorithms.

**Reachability,  $R_s$ :** It defines the ratio of successful navigation trials  $N_s$  over  $N_m$  attempts. A navigation trial where the rover succeeds in reaching the target without colliding with obstacles defines a successful navigation trial.

$$R_s = \frac{N_s}{N_m}, \quad (18)$$

**Computational complexity,  $T_c$ :** It defines the sum of the time the algorithm takes to compute each of the successful paths to the target until the rover reaches it.

**Path length,  $M_s$ :** It defines the total distance covered by the rover to reach its target. The average path length for  $N_s$  successful navigation trials is given by

$$\overline{M_s} = \frac{1}{N_s} \sum_{k=1}^{N_s} M_s(k), \quad (19)$$

with  $M_s(k)$  the path length of the  $k$ th successful run.

**Safety,  $SF$ :** Considering the robot has detected  $N$  obstacles during a successful navigation attempt  $k$  of length  $M_s$  at locations  $r_1, r_2, \dots, r_{M_s}$ , the safety parameter refers to the

average minimum distance between the robot and the detected obstacles. For  $N_s$  successful navigation trials it is defined as

$$\overline{SF} = \frac{1}{N_s} \sum_{k=1}^{N_s} \frac{\sum_{m=1}^{N(k)} d_{min}(r_o^m)}{N(k)}, \quad (20)$$

with  $d_{min}(r_o^m) = \min\{\|r_1 - r_o^m\|, \|r_2 - r_o^m\|, \dots, \|r_{M_s} - r_o^m\|\}$  and  $r_o^m$  the location of the  $m$ th detected obstacle. Safety will be used to compare field testing with simulation performance of the algorithms.

### C. Simulation results

Simulation results are listed in Table III. As anticipated, our reference algorithm, A\*, consistently reaches the target in every scenario. The classical APF approach exhibits poor results, reaching the target less than one in three instances as cluttering increases. RVF demonstrates relatively high reachability in non-cluttered environments but experiences a drastic decrease with cluttering. In contrast, CRBAPF\* maintains relatively high reachability independent of the environment. Notably, RAPF yields excellent results, exceeding CRBAPF\* reachability by more than 15%. Planning time for all the algorithms increases with cluttering (from Scenario A to Scenario C), with A\* having the longest computation time, as expected. Both RAPF and CRBAPF\* share the same planning time, outperforming both RVF and A\*. On average, they are 6 times faster than A\* and 2.5 times faster than RVF. RVF has a larger computation time than APF as it computes the field of forces and the rotation around each obstacles.

**Stability and outliers:** CRBAPF\* have a low variability but a certain amount of outliers, meaning that some configurations may lead to an abnormal computation time. RVF is the most stable and consistent algorithm in terms of planning time as it has a very low variability and almost no outliers. A\* presents the same characteristics as RAPF and CRBAPF\* in terms of outliers, but shows much higher median and variability, making it less consistent than RAPF. Outliers for CRBAPF\* come from the random walk technique implemented to escape the local minima, which our algorithm solves.

**Path length:** In terms of path length, the difference between algorithms becomes less evident. A\* and APF yield the shortest paths, but his result may be less reliable for APF, given that the mean path is computed based on successful paths. However, APF exhibits an overall low number of completed paths, indicating that path length is only computed in favorable environments. RVF produces paths between 5% to 10% longer than A\*. This can be attributed to RVF providing the same direction of rotation to the vector field of clusters of obstacles, leading to detours, while A\* finds a direct path between obstacles. CRBAPF\* yields a path length between 10% and 15% greater than A\*, attributable to the random walk technique implemented in CRBAPF\*. RAPF generates paths between 3% and 6% longer than those

TABLE III  
REACHABILITY, MEAN PLANNING TIME AND MEAN PATH LENGTH FOR EACH SCENARIO AND EACH ALGORITHM

Algorithm	Reachability [%]			Mean planning time [ms]			Mean path length [m]		
	Scenario A	Scenario B	Scenario C	Scenario A	Scenario B	Scenario C	Scenario A	Scenario B	Scenario C
A*	100	100	100	3906.6	5444.3	7699.1	38.6	38.5	38.7
APF	62.2	51.0	30.6	1225.8	1669.6	2180.0	<b>39.2</b>	<b>40.0</b>	<b>40.5</b>
RVF	73.6	63.4	46.6	1634.8	2237.5	2966.6	41	41.7	42.2
CRBAPF*	83	83.2	80	<b>579.6</b>	<b>888.1</b>	1328.6	43.5	44.2	45.2
<b>RAPF</b>	<b>96.4</b>	<b>93.8</b>	<b>91.8</b>	589.6	889.5	<b>1276.5</b>	39.8	40.3	41.1

computed by A\*, significantly lower than the CRBAPF\* paths. The path length for A\* remains independent of the environment, whereas RVF, RAPF, and CRBAPF\* exhibit a slight increase with cluttering. Path length distributions are highly asymmetrical. The median length is lower for CRBAPF\* than RVF, while the opposite holds for their means. This discrepancy arises from the substantial and widespread number of outliers generated by CRBAPF\*, attributed to the tail effect stemming from the random walk technique. This technique can produce extremely long paths as the path generation relies solely on randomness.

**Planning time:** In terms of planning time, A\* presents a large number of outliers but with low scattering. In contrast, RAPF displays half as many outliers and a considerably lower Interquartile Range (IQR) than CRBAPF\*, where the random walk technique is not used. The results suggest that RAPF is more consistent between paths, demonstrating increased stability and robustness. The RAPF algorithm displays approximately 50% lower navigation time, up to 200% higher success rate, and similar path lengths compared to the conventional APF algorithm. RAPF also outperforms RVF in success rate, path length, and planning time, while the planning time distribution closely mirrors that of the CRBAPF\*. Notably, it also outperforms the CRBAPF\* in path length, producing a near-optimal path.

#### IV. FIELD VALIDATION

##### A. Testbed description and testing approach

The rover is tested on a 3x4 m sandy terrain featuring rocks, craters, and slopes mimicking the lunar surface (see Fig. 1). The bottom left corner defines the origin of the 2D Cartesian reference frame. A Raspberry Pi 4B 2GB is used as an on-board computer on which the different path planning algorithms are implemented. Six motors drive the six legs. An IMU and an RF Tag are carried on board. The RF Tag communicates with 4 anchors located in each corner of the testbed for localization purposes. A threshold filter is applied to the raw data from the RF Tag to reduce noise. A Kalman filter merges the data from the motor encoders, IMU, and filtered RF Tag data at a frequency of 5Hz. Obstacles are detected with an on-board camera. Objects of a given size are used as obstacles to allow for the estimation of the distance and the angle from the robot. Obstacles are detected in a range of 0.8 m and a field-of-view of 62°. The software runs on Robotic Operating System (ROS) *Noetic*. Localization data,

which consists of the 2D coordinates of the rover ( $x_r$ ,  $y_r$ ) and its heading ( $\theta_r$ ), is shared with the path planning node. The distances, angles, and sizes of all detected obstacles are transmitted upon request from the planning node.

We completed a set of tests with different starting points, targets, and obstacle layouts. Arrangements that produce local minima are also tested to ensure the robustness of the RAPF algorithm. Tests have been conducted with up to 16 obstacles of varying sizes to comply with the lunar model presented before. Different terrain topologies were also tested (flat surface, overcoming small rocks, and slopes). Twenty tests were performed using virtual obstacle detection. This means that instead of using the on-board camera to detect obstacles, their positions are measured offline and provided to the planning algorithm. The detection is simulated in software as a perfect detection since all potential noise or uncertainty is removed. Simulated object detection assumes the same range and field-of-view of the on-board camera (i.e., 0.8 m, 62°). This enables us to evaluate the performance of the path planning algorithm irrespective of potential errors induced by other parts of the navigation pipeline. Ten tests were conducted using the on-board camera to prove that RAPF is robust to sensor-induced noise and uncertainties. Details of each test are provided in the following section.

##### B. Testing results and discussion

**Robustness:** In a first set of tests, we evaluated the robustness of RAPF against variations in the environment. Ten field tests are performed for two different obstacle layouts. These tests are performed with the virtual obstacle detection configuration only to avoid introducing further uncertainties into the analysis. Obstacles, starting points, and target layouts for the two scenarios, together with the computed and walked paths, are plotted in Fig. 2.

A t-test is performed to compare the ratio ( $\Omega$ ) of the actual walked path length over the computed path length for two different scenarios, to determine if the RAPF algorithm's behavior is influenced by the environment, with a null hypothesis suggesting no significant difference in the ratio of computed path length to actual walked path length between two map configurations. For a sample size of 10 for each group, a calculated t-statistic of 0.346 and a critical value of 2.110 for a two-tailed test on 17 degrees of freedom fails to reject the null hypothesis, indicating no statistical difference in the analyzed ratio.

The main challenge brought by real-world testing is the uncertainties. In a second set of tests, we evaluated the



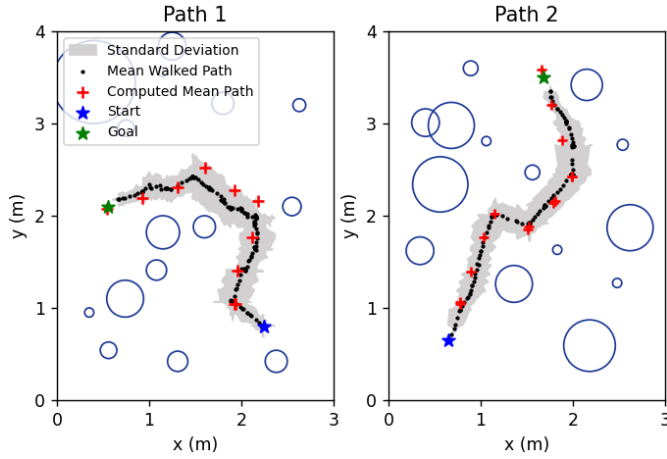


Fig. 2. Mean computed path (red), mean walked path (black) and standard deviation of the mean walked path (gray) for 10 repetitions for two different scenarios (Path1 & Path 2)

robustness of RAPF against sensor-induced uncertainties. In this case, data is collected for both rover configurations (virtual and on-board obstacle detection) and across Path 2. A Welch's t-test is employed with a sample size of 10 for each group, resulting in a calculated t-statistic of 0.64 and a degree of freedom of 15. With a p-value of 0.05 and a critical t-value of 2.131, the null hypothesis, suggesting no statistical difference in path length between the two scenarios, cannot be rejected as the calculated t-value is smaller than the critical value.

Lastly, it is worth highlighting that the minimum distance maintained between the rover and the nearest obstacle was 7cm on average when the on-board obstacle detection was used and 9cm when the virtual detection was used. The minimum distance kept from all obstacles was 20cm on average for the on-board detection and 26cm for the virtual detection. The hardware detection shows the lowest median and largest IQR, which means that when RAPF is implemented alongside on-board obstacle detection, the robot tends to travel closer to obstacles along its path. The on-board hazard detection leads to a smaller safety distance due to the greater variability in the estimation of the position of the obstacles relative to the rover.

**Computational complexity:** The path planning algorithm is run on the rover on-board computer, a Raspberry Pi 4B 2GB. The total computation time from the starting point to the target for the environments previously defined was always less than 0.2 s. The time to recompute one path took a maximum of 0.025 s.

For APF-based path planning methods, memory usage and CPU exploitation are intrinsically related to the size of the potential map. The smaller the map, the lower the overall computational resource needed to compute a path. In APFs, a map of the surroundings of the rover is stored in memory. Every time a new waypoint is reached, all path planning data is cleared from memory and new paths are recomputed. The RAPF algorithm is distinguished by the fact that at bacteria

points only the potential is computed. The number of planned paths depends on the level of cluttering of the environment as a new path is recomputed every time an obstacle is detected. For each new planned path, the number of potential values computed can be derived from:

$$N_{RAPF} = \frac{N_B M_s}{\rho_B}, \quad (21)$$

where  $N_B$  is the number of bacteria,  $M_s$  is the path length, and  $\rho_B$  is the distance between the rover and the bacteria point. The RAPF execution speed highly relies, therefore, on the distance between the rover and the targeted waypoint. Our field experiments showed that to reach a waypoint located about 3 m from the rover, the number of computed potential values for RAPF is 610 on average. In comparison, the estimated number of computed potential values for the grid-based APF and RVF algorithms would be between 1100 and 1300 for a  $5 \times 5 m^2$  stored map. Each obstacle is stored in 6 bytes. The size of the list of obstacles stored depends on the number of obstacles  $N_o$ . From the lunar model derived, up to 35 obstacles can be considered within a map of  $5 \times 5 m^2$ . The computed path to follow is also made of *float16* values. It consists of a list of coordinates  $(X_r, Y_r)$ , each couple having a size of 4 bytes. The total size depends on the length of the path to follow. For a  $5 \times 5 m^2$  and a waypoint located at around 3 m from the rover, RAPF approximately requires 0.5 KB of storage. In comparison, for storing a map of the same size, Field D\* needs around 10 KB of memory [22].

**Reachability:** The path planning module running onboard is capable of estimating the average time it would take to complete a traced path. In combination with the relative location of the rover and the current drawn by each motor, this can help the rover know if it is physically stuck or blocked. The algorithm knows the number of walking and turning steps required to reach a waypoint as well as the number of times a picture needs to be taken to check the path and the time needed to perform these tasks. These data are subject to slippage, actual resource allocation, and terrain uncertainties. Therefore a 100% margin is added to this estimated time. If the designated waypoint is not reached within the allocated time for executing the path, an error flag is raised and notified to the on-board computer. Data gathered by the rover while navigating are plotted in Fig.3. Potential maps as displayed in this figure are not stored on the rover and only computed for bacteria points; only the list of obstacles is stored on board. The potential maps plotted here are for environmental visualization purposes only. Regions between the obstacles and  $\rho_0$  normally have infinite potential, represented in these plots by a very large finite value. On the left image, no local minima escape method is implemented (i.e., CRBAPF path planner). The rover gets stuck at the first local minimum encountered. On the right image, the RAPF algorithm is used, which implements a potential function assigned to local minima as described before. The path leading to the local minimum is cleared and a new path is computed considering the potential generated by the local minimum. This leads to

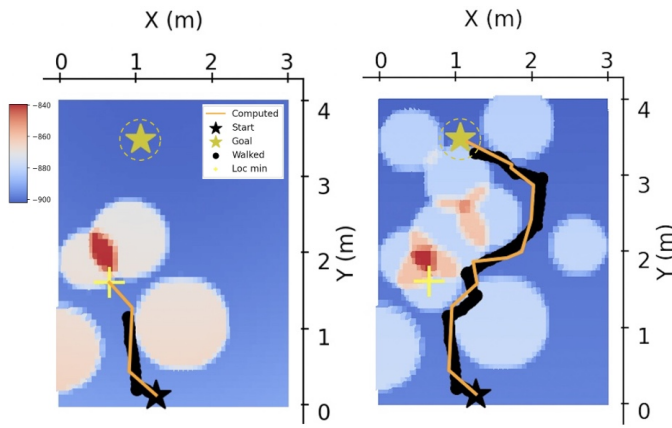


Fig. 3. Potential map, computed path (orange), and path followed by the rover (black). Left: CRBAFF. Right: RAPF

a new path capable of reaching the target, as can be seen in the figure. In some cases, the algorithm does not manage to plan a path that avoids the local minimum despite the local minima technique implemented. In this case, an error flag 'local\_minimum' is set by a timer. If no local minimum-free path is output after 10 seconds, the path planning algorithm notifies the OBC that no path can be found. This flag prevents the rover from continuing to search an impossible path and avoids unnecessary resource consumption.

## V. SUMMARY

The RAPF algorithm enables real-time computation of new paths for small and resource-limited mobile robots. The proposed approach drastically increases the success rate compared to state-of-the-art APF methods by effectively addressing the local minima issue. The bacterial approach of the RAPF algorithm, minimizes memory requirements, making it ideally suited for robotic systems with strong computational limitations. Simulation results show a 200% higher success rate and 50% lower computing time than the classical APF algorithm, with only a 5% longer path length than the optimal algorithm,  $A^*$ , in a lunar-like environment. It also outperforms state-of-the-art APF-based algorithms by more than 15% in reachability and 10% in path length for a similar or shorter planning time. During the field validation, the RAPF algorithm displayed robust performance, effectively adapting to uncertainties introduced by various sensors. The RAPF algorithm performs well in various situations. The rover walks in a straight line towards the target when no obstacles are detected. It navigates smoothly around small discrete obstacles. The path planning performance of the rover do not depend on the relief of the terrain. Its consistent performance across diverse environments underscores its capability for real-time autonomous navigation over challenging terrains.

## ACKNOWLEDGMENT

We express our appreciation to Lunar Zebro [23] for providing the rover prototype and granting access to their testing facility. The project is partially funded by the Moonshot program supported by the Delft Space Institute.

## REFERENCES

- [1] X. Yu, Q. Huang, P. Wang and J. Guo, "Comprehensive Global Path Planning for Lunar Rovers," 2020 3rd International Conference on Unmanned Systems (ICUS), Harbin, China, 2020, pp. 505-510
- [2] Peng, Q., Wang, S., Zhang, J., "Fast Lunar Path Planning Based on Intelligent Heuristic Map," International Conference on Autonomous Unmanned Systems 2022.
- [3] Dijkstra, E., "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, 1959
- [4] P. Hart, N. Nilsson, B. Raphael, B., "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, pp. 100-107, 1968
- [5] LaValle, S., "Rapidly-exploring random trees : a new tool for path planning," *The annual research report*, 1998
- [6] L. Kavraki, P. Svestka, J. Latombe, M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, 1996
- [7] S. Karaman, E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning", *International Journal of Robotics Research*, 2011
- [8] W. McCulloch, W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics* 5, 115-133, 1943
- [9] J. Kennedy, R. Eberhart, "Particle swarm optimization", *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, pp 1942-1948, 1995
- [10] M. Dorigo, M. Birattari, T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, pp. 28-39, 2006
- [11] S. Katoch, S. Chauhan, V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimed Tools Appl* 80, 8091-8126, 2021
- [12] Holland, John H. "Genetic Algorithms." *Scientific American*, vol. 267, no. 1, 1992, pp. 66-73. JSTOR, <http://www.jstor.org/stable/24939139>. Accessed 16 Jan. 2024.
- [13] Khatib, O., "Real-time obstacle avoidance for manipulators and mobile robots," *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, pp 500-505, 1985
- [14] L. Nasuha, L. Priambodo, G. Pratama, "Vortex artificial potential field for mobile robot path planning," *Journal of Physics: Conference Series*, vol. 2406, 2022.
- [15] Gelling, L., "A lunar surface path finding and obstacle avoidance algorithm," M.S. thesis, Delft University of Technology, 2023.
- [16] A. Hossain, I. Ferdous, "Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique," *Robotics and Autonomous Systems*, vol. 64, pp. 137-141, 2015.
- [17] Ahmad, Y. "Matlab implementation of artificial potential field," <https://github.com/Yaaximus/artificial-potential-field-matlab>
- [18] M. Diab, M. Mohammadkarimi, R. Rajan, "Artificial potential field-based path planning for cluttered environments," *IEEE Aerospace Conference*, 2023, pp. 1-8
- [19] M. Golombek, D. Rapp, "Size-frequency distributions of rocks on Mars and Earth analog sites: Implications for future landed missions," *Journal of Geophysical Research: Planets*, vol. 102, pp. 4117-4129, 1997
- [20] M. Golombek, A. Haldemann, N. Forsberg-Taylor, E. DiMaggio, R. Schroeder, B. Jakosky, M. Mellon, J. Matijevic, "Rock size-frequency distributions on Mars and implications for Mars Exploration Rover landing safety and operations," *Journal of Geophysical Research: Planets*, vol. 108, 2003
- [21] Y. Li, B. Wu, "Analysis of Rock Abundance on Lunar Surface From Orbital and Descent Images Using Automatic Rock Detection," *Journal of Geophysical Research: Planets*, vol. 123, 2018
- [22] J. Carsten, A. Rankin, D. Ferguson, A. Stentz, "Global Path Planning on Board the Mars Exploration Rovers", *IEEE Aerospace Conference*, 2007, pp. 1-11
- [23] R.T.Rajan, A.Noroozi, P.Sachdeva, A.Menicucci, C.Verhoeven, "Lunar Zebro - An Autonomous Moon Rover", 75th International Astronautical Congress, 2024