A COOPERATIVE UGV-UAV PATH PLANNING ALGORITHM IN $\mathbb{R}^3$ SPACE FOR PLANETARY EXPLORATION

F. Ropero[1], **J. Caballero**[1], **P. Muñoz**[1], **M. D. R-Moreno**[1,2], **and H. Hildmann**[2]

[1]*Universidad de Alcalá, Escuela Politécnica Superior, ISG, Alcalá de Henares, Madrid, Spain*
[2]*Dutch organization for applied scientific research (TNO), IAS, The Hague, The Netherlands*

ABSTRACT

In this paper, we consider the scenario of exploring a planetary surface with a system formed by an Unmanned Aerial Vehicle (UAV) and an Unmanned Ground Vehicle (UGV). The goal is to reach a set of target points minimizing the travelling distance. On previous work, we presented the TERRA algorithm, which was able to solve this type of problems in an $\mathbb{R}^2$ Euclidean space. This time, we explain an extension that allows TERRA to deal with $\mathbb{R}^3$ instances, resulting in 3DTERRA, and to apply our work to exploratory missions. We will describe the algorithm(s) modifications and test this new version in new 3D environments, testing how the novel paradigm affects previous results. The experimental evaluation of 3DTERRA demonstrates that its performance relies on the battery capacity of the UAVs, the clustering level of the target points in the exploration area, and the complexity of the terrain features.

Key words: Exploration; Cooperation; Routing; Heterogeneous Robots.

1. INTRODUCTION

The state of the art with regard to autonomous or semi-autonomous vehicles has rapidly progressed in the last years. Most of the attention received by the media and the mainstream public is, unsurprisingly, focused on consumer products (such as e.g., UAVs or drones) and vehicles for personal transportation (e.g., TESLA). In the military and industrial sector, robotics has also made significant advances with regard to mobile systems capable of operating at least semi-autonomously. Specifically, the mining industry is increasingly investing in more and more advanced research to enable autonomous operation under ground, an environment where communication is notoriously difficult and where positioning, navigation and timing (PNT) is a crucial challenge. Various recent research projects have investigated cooperative formations of multiple, individually operating vehicles. Some of the advantages of operating a collective (often referred to as a Swarm) of vehicles, as opposed to a single physical system, are immediately apparent: in dangerous environments, loss of a single unit does not constitute a loss of the entire system; multiple systems can cover a larger physical area; distributed systems can offer self-healing and re-calibration capabilities that single systems lack.

The rapidly growing space industry sector shares many of the challenges of e.g., under ground mining. While there may not be anything physically obstructing the communication, extremely large distances can incur a communication delay that makes remote controlling a device impractical or even impossible. Therefore, device autonomy as well as multi-device collaboration are of interest to the stakeholders in this domain. One of the more recent missions highlighting this was the deployment of the Mars Helicopter Scout (MHS) along with the Mars 2020 Rover [NAS20], which received widespread media attention. Space agencies, such as the National Aeronautics and Space Administration (NASA) or the European Space Agency (ESA), are defining new cooperation paradigms aiming to improve the quality and quantity of the scientific return from their exploration missions. One of the most studied problems is the area coverage using multiple Unmanned Ground Vehicles (UGVs) [BMSS05] or Unmanned Aerial Vehicles (UAVs) [RBTH02].

Furthermore, one of the most pressing bottlenecks for operation in space as well as on other planets is resource depletion and power consumption. Algorithms optimizing power consumption and improving operational efficacy are in high demand as they can extend mission times, push the boundaries of what a multi-robot system can possibly achieve and generally make these systems more versatile and robust once they are deployed.

In previous work [RMRM19, Rop20] we have reported on a novel path planning algorithm for heterogeneous multi-robot systems, specifically for UAVs operating from a mobile (and moving) UGV. Our work was directly cast in the scenario of a rover operating autonomously for a dynamically changing task and using UAVs to augment its capabilities. Prior art shows a broad diversity of cooperative UGV/UAV exploration systems, but with regard to high-level autonomous explorations (were both UGV and UAV are fully autonomous systems) our work constituted a contribution to the state of the art. Since the previous reports, our work has progressed further.

In this article we will revisit the original work, which was developed for 2D environments / enabled the cooperative system of systems to be operating basically as a ground based vehicle. We will extend the results to 3D environments and apply our work to exploratory missions.

Section 2. Next section explains the exploration problem we aim to solve. Section 3 will describe the original TERRA algorithm, as we will build upon it in the next sections. Section 4 explains the steps solve $\mathbb{R}^3$ instances, resulting in the 3DTERRA algorithm. Section 5 shows experimental results where we demonstrate the effectiveness of the new algorithm in $\mathbb{R}^3$ environments. Finally, conclusions are outlined.

2. THE PROBLEM DEFINITION

The exploration problem is called the *Energy Constrained UAV and Charging Station UGV Routing Problem* (ECU-CSURP) [RMRM19, Rop20]; it is defined as:

(i) an $\mathbb{R}^2$ Euclidean space $\mathtt{A}$: the exploration area.

(ii) a set $\mathcal{L}$ of locations l_j and a set of target locations $\mathcal{T}$ (with $\mathcal{T} \subset \mathcal{L}$) within $\mathtt{A}$. One of the locations, l_0, is the *home* location. Each location l in this $\mathbb{R}^2$ space is defined by its coordinates: (x_l, y_l).

(iii) a simple heterogeneous multi-robot (UGV-UAV) system. Both of these systems are modelled as Dubins vehicles [Dub57], i.e., vehicles that can only move forward and are doing so at a constant speed.

with the following constraints:

(iv) the maximum distance any UAV can travel on a single battery charge is limited (energy constraint).

(v) at least one location l_x is further away from l_0 than the UAVs' energy constraint allows them to travel.

Contrary to the UAV(s), the UGV is not subjected to an energy constraint and can move freely. The UGV can furthermore serve as a charging station for the UAV, meaning that UAV(s) that have landed on the UGV are being recharged to their maximum travel distance. In our model we consider a UGV-UAV system with only one UAV.

The objective of the ECU-CSURP is to find a cooperative routing for the simple UGV-UAV system that (a) enables the UAV to visit every target in $\mathcal{L}$ (starting from, and returning to, l_0) while trying to minimize the overall traveling distance.

In our model, we assume constant velocities for both the UGV (v_{UGV}) and the UAV (v_{UAV}). For a constant battery capacity this means that there is a maximum flight time $t_{\max}$ and thus a maximum distance $d_{\max}$ a UAV can travel on one battery charge. Since the UAV has to return to the UGV to recharge, the operational radius r of the UAV can be calculated as $d_{\max}/2$ (assuming that the UGV does not move). For significant differences in speed for UGVs and UAVs (orders of magnitude) we assume that this simplification suffices to estimate r (as shown in Figure 1).

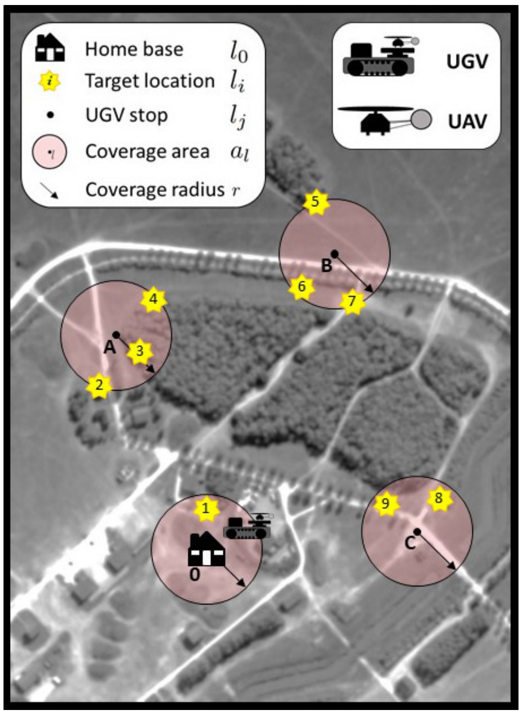


Figure 1: An ECU-CSURP problem with 9 target locations l_1,..., l_9 plus l_o, the home base of the UGV-UAV system. r (half the maximum flight distance of the UAV) defines a_l ($a_l \subseteq \mathtt{A}$), the area the UAV can safely cover when taking off from location l. In the above example, the 4 locations l_0, l_A, l_B and l_C (i.e., UGV stops) suffice to position the UAV within reach of all 9 target locations.

3. THE TERRA ALGORITHM ($\mathbb{R}^2$)

Before presenting the revised algorithm for $\mathbb{R}^3$ spaces we first introduce the original: TERRA, the cooperaTive ExploRation Routing Algorithm, a cooperative path planning algorithm to solve the ECU-CSURP for $\mathbb{R}^2$ spaces.

TERRA enables the cooperation between two classes of autonomous / unmanned vehicles and capitalizes on the resulting synergies: the larger and much slower UGV, while restricted by the terrain in its movement, serves as a mobile charging station for the much faster airborne UAVs. However, other than the UAV, a UGV incurs little or no operational cost when stopped outside the home base and can therefore operate much longer (in time) than a UAV. In addition, it can also slowly move through the environment while resupplying a UAV with power for another flight, thereby moving the coverage area for the UAV (which has the UGV as its center; cf. Figure 1).

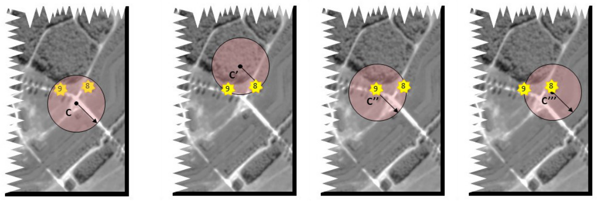
Given a set of target locations, TERRA determines a favorable sequence of waypoints (charging stops) that, when traversed by the UGV, enable the UAV to fly to the target locations that are within reach before returning to the UGV in order to recharge and be transported to the next waypoint, and, ultimately, back home.

To achieve this, the TERRA algorithm uses five stages:

1. Voronoi search:
   Calculating stopping positions for the UGV. The UAV has a maximum field of coverage. The Voronoi search essentially looks for locations that can serve as the center of such coverage circles (UGV stops) while including as many target locations as possible. The computational methods follows a similar approach to the Preparata and Shamos's method [PS12].

2. Combinatorial optimization:
   Finding an optimal set of such UGV stops, so that all stops together cover all target locations, and do so with as few circular areas as possible. In e.g., Figure 1 only 3 stops (as well as the initial position at the home base) suffice to bring all target locations within the coverage of the UAV.

3. Gravitational optimization:
   With a potentially large number of locations (centers for coverage) equally capable of covering a given set of target locations, there is room for further optimization. See Figure 2 for an example.



l_C (cf. Fig. 1) Alternate options for l_C

Figure 2: The UGV stop l_C from Fig. 1 and example alternates, which cover the same 2 target locations. Option 1 has the UGV stop in the forest, while options 2 and 3 are basically identical with one of the target locations.

   We assume that we optimize for operational time. Since the UGV is significantly slower than the UAV we prefer UGV stops that are closer together (i.e., closer to their combined center of gravity).

4. UGV path heuristic using a Genetic Algorithm:
   Having a minimal set of UGV stops, even when these are as close together as possible does not yet provide us with a path for the UGV. Finding an optimal path (the well-known *Travelling Salesman Problem* (TSP) [Cum]) is solved by a Genetic Algorithm applying a tournament selection method [GD91] to repeatedly select the best individual of a randomly chosen subset.

5. An A^* [HNR68] based heuristic for the UAV path:
   Similar to the UGV path, the UAV path can be optimized for minimal length. This differs slightly from the problem for the UGV as the UAV may have to return to charge even within a single field of coverage if there are too many target locations. This means that the TSP from above changes in that it allows the UGV to be visited multiple times. We solved this with a variation of the A^* algorithm.

Stage 1: Voronoi search

This algorithm takes a set $\{l_i, \ldots, l_{i+n}\}$ of target locations as input and partitions $\mathbb{A}$ into areas such that (a) the union of all of these areas is an area inside $\mathbb{A}$ that contains all target locations. The output is a set of such areas, defined not by a location but, abstractly, by sets of target locations that can be placed inside an area with radius r.

Input: *TargetLocations*: $\{l_i, \ldots, l_{i+n}\}$
Output: *Custers*$_{all}$

So each cluster is a subset of the target locations and the union of these sets equals the set of all target locations.

Stage 2: combinatorial optimization

In stage two, the partitioning of the target locations into clusters of target locations, provided by stage 1, is optimized: the algorithm returns the minimal set of clusters that covers all target locations.

Input: *Clusters*$_{all}$
Output: *Clusters*$_{min}$

with *Clusters*$_{min} \subseteq$ *Clusters*$_{all}$.

Stage 3: gravitational optimization

In this stage, the clusters (so far only defined abstractly as the set of target locations which they cover) are mapped to specific locations in $\mathbb{A}$. The algorithm determines, for each cluster, a suitable center location (i.e., the position for the UGV stop). It then optimizes these locations such that each UGV stop is moved as close as possible to the center of gravity of the set of locations (while still keeping its target locations inside the coverage of that stop).

Input: *Clusters*$_{min}$: $\{c_j, \ldots, c_{j+m}\}$
Output: *ChargingStops*: $\{l_{c_j}, \ldots, l_{c_{j+m}}\}$

Stage 4: a GA heuristic for the UGV path planning

We developed a genetic algorithm based heuristic to determine the shortest path for the UGV.

Input: *ChargingStops*: $\{l_{c_j}, \ldots, l_{c_{j+m}}\}$
Output: $\text{path}^{\text{UGV}}_{\{l_{c_j}, \ldots, l_{c_{j+m}}\}}$

Stage 5: a heuristic for the UAV path planning

Finally, for each charging stop for the UGV, we calculate the shortest path (or paths) that enable the UAV to visit all target locations. Since we may have to revisit the UGV for intermittent charging of the UAV, an approach is needed that can return multiple paths.

Input: $\mathit{Clusters}_{min}$: $\{c_j, \ldots, c_{j+m}\}$ and *ChargingStops*: $\{l_{c_j}, \ldots, l_{c_{j+m}}\}$
Output: $\text{path}^{\text{UAV}}_{\{c_j \cap l_{c_j}\}}, \ldots, \text{path}^{\text{UAV}}_{\{c_{j+m} \cap l_{c_{j+m}}\}}$

Once stage 5 is completed we have all the information needed to send the UGV-AUV system on its way. The interested reader is referred to [RMRM19, Rop20] for a detailed discussion of each of the mentioned algorithm as well as extensive performance evaluation thereof.

4. THE 3DTERRRA ALGORITHM ($\mathbb{R}^3$)

UGVs can operate in 3 dimensions and many interesting and relevant application scenarios feature (often significant) elevations and obstacles. Equipping the algorithm to handle $\mathbb{R}^3$ topologies is therefore the logical next step.

The main contribution of this paper is the extension of the TERRA algorithm to 3D spaces. First, we extended the ECU-CSURP from $\mathbb{R}^2$ to $\mathbb{R}^3$ Euclidean spaces (§4.1). We then briefly introduce a model for such spaces that allows us to determine whether a given location is reachable by our UGV (§4.2) and then provide the Voronoi algorithm augmented for $\mathbb{R}^3$ Euclidean spaces (§4.3).

4.1. Extending the problem from $\mathbb{R}^2$ to $\mathbb{R}^3$ space

The first step, unsurprisingly, is to extend the coordinates for any (2D) location l from (x_l, y_l) to a 3D representation: (x_l, y_l, z_l). It should be noted that this change only affects stages 1 and 3 (the Voronoi tesselations (Stage 1) to determine suitable clusters of location as well as the gravitational optimization (Stage 3) of a specific set of UGV stops, as the calculations for both stages are performed on a 3D map. The remaining stages (Stages 2, 4, 5) only require calculations that are dimension agnostic, using e.g., a distance matrix where the values are now simply calculated over 3 dimensions as opposed to two.

It should be noted that only the ground based vehicle (UGV) is affected by the added dimension: a rover will have to cope with uneven terrain by traversing areas of varying inclination. The UAV can simply increase in altitude and fly over unsuitable terrain. This motivates our decision to continue to model UAVs in only two dimensions (the equivalent to flying at a constant altitude).

4.2. The Digital Terrain Model (DTM) for $\mathbb{R}^3$ space

To model $\mathbb{R}^3$ space we chose to adhere to the Digital Terrain Model (DTM), a well used 3D computer graphic representation of $\mathbb{R}^3$ Euclidean spaces. A DTM represents the bare ground surface (elevations) but without any objects like trees or buildings (obstacles). With the domain of planetary exploration on e.g., Mars in mind, we argue that focusing primarily on elevation is justified.

Muñoz et al. [MRMC17] model a DTM as a grid of rectangular cells, where each cell is formed by four adjacent nodes on the map. These four nodes, combined with a central point of the cell, form four triangles. The slope of each triangular plane determines the slope of each node at the rectangular cell. This model expresses this computed slope and other terrain characteristics as a numerical value. This value indicates the cost (i.e., the estimated effort) required to cross an area on the map.

When imposing a threshold on this cost we can distinguish two kinds of nodes, namely:

- Legitimate (reachable) nodes, i.e., nodes that can be crossed by the UGV (because the projected cost its lower than a given, UGV-dependent threshold) and represent a feasible option for a charging stop; and
- Illegitimate (unreachable) nodes, that cannot be crossed by the UGV due to prohibitively high costs.

We use the DTM geometric formulation presented in Muñoz et al. [MRMC17] to compute the objective functions of the amended ECU-CSURP. It defines a method to compute the approximate distance travelled by a vehicle through adjacent (or not adjacent) 3-dimensional nodes. For that, they interpolate the elevation of nodes that are inside of the rectangular cell. To use this, we only have to extend the original problem definition's coordinates for any (2D) location l from (x_l, y_l) to a 3D representation: (x_l, y_l, z_l), with z_l the elevation obtained from the DTM.

4.3. Extending the Voronoi search from $\mathbb{R}^2$ to $\mathbb{R}^3$

As the TERRA's first stage, our Voronoi's search method looks for placing intermediate charging stops in the UGV's path, so the UAV can reach the target points without running out of energy. In $\mathbb{R}^2$ Euclidean spaces, the map is an open (without obstacles) and flat terrain, so every point is a legit node (also called as legit vertex) to be a charging stop, because the UGV can cross every point without problems. Nevertheless, as we mentioned in the previous section, a point may not be legit to be crossed by an UGV in DTMs.

We updated our Voronoi search algorithm as outlined in Alg. 1. This revised algorithm differs from the original version (described above, for full details the reader is referred to [RMRM19]) with the three main changes being:

1. The computed Voronoi vertices in the Voronoi function (line 10 of Algorithm 1) are formed by legit vertices $\mathcal{L}^{\text{UGV}}_{\text{Reachable}}$, illegit vertices $\mathcal{L}^{\text{UGV}}_{\text{NonReachable}}$ and vertices which are out of the map boundaries. This distinction is mandatory in order to select legitimate vertices that cover the target points, and the nearest vertices of each target point not covered yet (line 24).

2. The f_{dist} function (line 17) now computes the 3-dimensional (as opposed to 2D) distance between a target point and a legit vertex using Pythagoras.

3. The f_{dummy} function (line 8) computes the false Voronoi vertex (v_{dummy}) to facilitate covering an isolated (lone) target point. This is necessary because the Voronoi function cannot compute a Voronoi tessellation with only two vertices. The f_{dummy} function roams around the target point in a spiral, starting at r and moving inwards. The goal is to find a legit vertex inside the coverage area.

Furthermore, the algorithm requires the cost map ($\mathbf{Map}$), which is exported from the DTM and operates on the following sets: $\mathcal{T}_{\text{COVERED}}$, the set of target locations that are already covered, $\mathcal{T}_{\text{OPEN}}$, the set of target locations that are not yet covered ($\mathcal{T}_{\text{COVERED}} \cap \mathcal{T}_{\text{OPEN}} = \mathcal{T}$, the initial set of target locations). The algorithm outputs a set of charging locations for the UGV, $\mathcal{L}_{\text{UGV}}$. The algorithm furthermore makes use of $\mathcal{L}^{\text{near}}_{\mathcal{L}'}$, the set of all neighbours for any location in some set $\mathcal{L}'$, calculated by function f_{near}. As discussed above in §4.2 we use the DTM to distinguish reachable from unreachable locations. Internally, these are calculated by $f_{\mathcal{L}^{\text{UGV}}_{\text{Reachable}}}$ and $f_{\mathcal{L}^{\text{UGV}}_{\text{NonReachable}}}$ and represented by $\mathcal{L}^{\text{UGV}}_{\text{Reachable}}$ and $\mathcal{L}^{\text{UGV}}_{\text{NonReachable}}$. The Voronoi algorithm, $f_{Voronoi}$, takes a set of target locations $\mathcal{T}$ as input and returns calculated Voronoi tesselations $\mathcal{V}$.

The introduction of legitimate and illegitimate vertices does not only impact the Voronoi search, but also the gravitational optimization algorithm. In this way, its algorithm is updated to detect if the computed junction point is a legit or an illegit vertex. That is, the new algorithm works in a same way but computing the corresponding $\mathbb{R}^3$ Euclidean functions.

Computing 3-dimensional UGV's paths represents a significant enhancement of the TERRA algorithm to solve $\mathbb{R}^3$ problem instances. However, we do not compute the UAV's 3-dimensional path as well as other physical constraints, such as wind speed or atmospheric density for two reasons: first, some of them imply an explosion in the problem complexity and, second, dynamic constraints (as wind speed) cannot easily modelled neither predicted to be exploited in off-line planning (they are better suitable for on-line adaptation of the plan during execution).

For the UGV's 3-dimensional path, we introduce an additional stage after computing the TSP for the two-dimensional UGV's path (Stage 4). This new stage is the 3Dana algorithm [MRMC17] a path planning algorithm developed to obtain safer routes based on heuristic search over a DTM and/or a traversability cost map.

Algorithm 1 3D Voronoi search method

```
1: Procedure Voronoi3DSearch($\mathcal{T}, r, Map$)
2: $\mathcal{T}_{\text{OPEN}} \leftarrow \mathcal{T}$
3: $vertices \leftarrow \mathcal{T}_{\text{OPEN}}$
4: $l, \mathcal{V}, \mathcal{L}^{\text{near}}, \mathcal{L}^{\text{UGV}}_{\text{Reachable}}, \mathcal{L}^{\text{UGV}}_{\text{NonReachable}} \leftarrow \emptyset$
5: while $\mathcal{T}_{\text{OPEN}} \neq \emptyset$ do
6:     $v_{dummy} \leftarrow \emptyset$
7:     if $|vertices| == 2$ then
8:         $v_{dummy} \leftarrow f_{dummy}(vertices), r, Map)$
9:     else
10:        $\mathcal{V} \leftarrow f_{Voronoi}(vertices)$
11:        $\mathcal{L}^{\text{UGV}}_{\text{Reachable}} \leftarrow f_{\mathcal{L}^{\text{UGV}}_{\text{Reachable}}}(\mathcal{V}, Map)$
12:        $\mathcal{L}^{\text{UGV}}_{\text{NonReachable}} \leftarrow f_{\mathcal{L}^{\text{UGV}}_{\text{NonReachable}}}(\mathcal{V}, Map)$
13:    end if
14:    $\mathcal{L}^{\text{UGV}}_{\text{Reachable}} \leftarrow \mathcal{L}^{\text{UGV}}_{\text{Reachable}} \cup v_{dummy}$
15:    for $l \in \mathcal{T}_{\text{OPEN}}$ do
16:        for $l \in \mathcal{L}^{\text{UGV}}_{\text{Reachable}}$ do
17:            if $f_{dist}(t, l) < r$ then
18:                $\mathcal{T}_{\text{OPEN}} \leftarrow \mathcal{T}_{\text{OPEN}}/l$
19:                $\mathcal{L}_{\text{UGV}} \leftarrow \mathcal{L}_{\text{UGV}} \cup l$
20:            end if
21:        end for
22:    end for
23:    for $l \in \mathcal{T}_{\text{OPEN}}$ do
24:        $\mathcal{L}^{\text{near}} \leftarrow \mathcal{L}^{\text{near}} \cup f_{near}(l, \mathcal{L}^{\text{UGV}}_{\text{Reachable}}, \mathcal{L}^{\text{UGV}}_{\text{NonReachable}})$
25:    end for
26:    $vertices = \mathcal{T}_{\text{OPEN}} \cup \mathcal{L}^{\text{near}}$
27: end while
28: $A \leftarrow ComputeAreas(V, r)$ return $\mathcal{L}_{\text{UGV}}, A$
29: End Procedure
```

The 3Dana algorithm generates long term paths, exploiting a DTM geometric formulation without requiring a mechanical model of the robot. Due to this, the paths generated are safer than the ones obtained from representations combined into a single cost map.

5. EXPERIMENTS

To evaluate performance two experiments were devised: one assesses the ability of the 3DTERRA algorithm to find the minimal set of legit locations for the UGV and the second evaluates the overall performance of 3DTERRA.

The first is necessary because the search for legit locations is costly but can reduce the effectiveness of stage one and two of the algorithm, if done sub-optimally, especially in rugged terrains. To address this, a clustering optimization was performed to assess the algorithms performance for finding the minimal set of locations required to cover the target locations. The second is related to the integration of 3Dana [MRMC17] into the algorithm.

Both experiments have been performed under a real Mars DTM, namely the central uplift of a 30-Km diameter crater in Noachis Terra, as captured by the High Resolution Imaging Science Experiment (HiRISE) on board the Mars Reconnaissance Orbiter. Its identification in the HiRISE database is: DTEED-030808-1535- 031230-1535-A01 [Uni]. Taking advantage that the 3Dana algorithm aims to find safe routes for the UGV, we wanted to assess the performance of these experiments according to different safety levels in the navigation of the heterogeneous simple UGV-UAV system. These safety levels allows us to describe several risky environment settings through the next two parameters:

- Terrain slope (P): 3Dana uses terrain slope as a threshold to compute a UGV's path where every point has less slope than this threshold. A high safety level will be determined by a lower slope as the UGV's path avoids dangerous terrain features.
- Security range (β): the distance percentage that is going to be subtracted from a theoretical farthest distance (r_{max}) the UAV can travel without running out of energy, e.g., with $r_{\text{max}} = 300$ meters and $\beta = 0.1(10\%)$. Then, $r = r_{\text{max}} - (r_{\text{max}} * \beta) = 270$ meters. The goal of β is to avoid depleting the UAV's battery due to unpredictable conditions.

5.1. Clustering optimization experiment

The first experiment evaluates the first and second stages of 3DTERRA. We have described that the goal of these stages is to minimize the number of legit vertices, denoted as $\mathcal{L}^{\text{UGV}}_{\text{Reachable}}$, required to cover the whole set of target points. But, we cannot ensure that $\mathcal{L}^{\text{UGV}}_{\text{Reachable}}$ is optimal on every distribution. Then, we use a random scenario generator of to define the optimal number of legit vertices, known as δ, on every distribution.

The random map generator creates every scenario ensuring that $\mathcal{T}$ target points are distributed in δ legit vertices, inside a specific radius R and allocated around a real Mars DTM. Here, the δ parameter allows us to define the optimal number of legit vertices to cluster every target point on each scenario, i. e., 3DTERRA can compute, at the very least, δ legit vertices. Therefore, we are able to compare $\mathcal{L}^{\text{UGV}}_{\text{Reachable}}$ and δ to detect when 3DTERRA has computed an optimized scenario. We have defined $\mathcal{L}^{\text{UGV}}_{\text{Reachable}} = \delta$ as an optimized scenario.

This experiment consists of the execution of ten thousand random scenarios over the Mars DTM with the four different safety levels based on r and P (L1 = low safety, L4 = high safety) displayed in Table 1. P is ranged from 20º to 1º (plain terrain). β goes from 10% (low security range) to 90% (high security range). Then, the goal of this experiment is to determine the optimized scenarios percentage in the L1-L4 safety levels.

The results in Table 1 show that as long as the safety level increases from L1 to L4, the percentage of optimized scenarios (when $\mathcal{L}^{\text{UGV}}_{\text{Reachable}} = \delta$) decreases. L1 shows the highest percentage of optimized scenarios, because a high P and a low β represents more possibilities to find legit vertices around the target points, and then, to achieve $\mathcal{L}^{\text{UGV}}_{\text{Reachable}} = \delta$. L4 shows the lowest percentage of optimized scenarios, because of the highest β and the lowest P, so there are few legit vertices around the target points.

Also, we can appreciate that the computational time increases as long as it does the safety level, because 3DTERRA finds less legitimate vertices to ensure legit vertices lose to the target points. So, if a legit vertex cannot be reached by the UGV, its means that the target point cannot be visited by the UAV and then, the scenario does not have solution. As shows the L4 results, 3DTERRA does not find a solution for 2759 scenarios.

5.2. Computational performance vs. Safety

The second experiment evaluates the performance of 3DTERRA as a whole, over different safety levels. The performance has been assessed in terms of the computational time taken to solve every scenario and the computed distance travelled by the UGV-UAV system.

This experiment consists of the evaluation of one hundred scenarios over the Mars DTM [Uni] with the safety levels displayed in Table 2.

P is ranged from 20º to 5º (almost a plain terrain). β goes from 10% (low security range) to 90% (high security range). The results in Table 2 shows that as safety levels increase from L1 to L4, the travelling distance clearly grows, the number of scenarios with solution decreases and the computational time increases exponentially.

The distance travelled by both robotic systems is the lowest for safety level L1 since the UGV has more ability to avoid obstacles (high P) and the UAV can fly a farther distance (low β). Then, it is easier to reach a legit vertex close to a target point, which in turn minimizes the distance travelled by the robots. Also, the computed scenarios with solution decreases in a higher safety level due

Table 1: Clustering optimization of the first two 3DTERRA's stages in different safety levels over a Mars DTM [Uni]. It has been computed a total of 10000 random scenarios for each safety level. `SS` = scenarios with solution, `WS` = scenarios without solution, and `OS` = optimized scenarios. RTime is the runtime in milliseconds.

Level	$\beta(\%)$	P (º)	#`SS`	#`WS`	`OS` (%)	RTime (ms)
L1	10	20	10000	0	71.0	6.5
L2	30	10	10000	0	62.3	12.2
L3	60	5	10000	0	63.1	16.7
L4	90	1	7241	2759	39.2	26.1

Table 2: Performance of 3DTERRA under different safety levels over a Mars DTM [Uni]. Every level has been tested over the same 100 random scenarios. F_{ugv} = distance travelled by the UGV in km, F_{uav} = cumulative distance travelled by the UAV, SS = scenarios with solution, and WS = scenarios without solution. RTime is the runtime in miliseconds.

Level	$\beta(\%)$	P (º)	$F_{ugv}(km)$	$F_{uav}(km)$	#SS	#WS	RTime (ms)
L1	10	20	10679.2	1096.1	93	7	226.6
L2	30	15	10727.5	1095.7	65	35	485.6
L3	60	10	11807.8	1109.7	4	96	3562.0
L4	90	5	0	0	0	100	0

to the UGV has less ability to avoid obstacles (low P) and the UAV can fly a shorter distance (high β). Then, it is more difficult to reach a legit vertex. Consequently, the computational time increases because of the UGV's path planning computed by the 3Dana algorithm.

6. CONCLUSIONS

We have explained an extension to the TERRA algorithm [RMRM19] that allows for $\mathbb{R}^3$ ECU-CSURP instances solving. This new problem modelling permits our algorithm to be applied in new exploratory scenarios that better resemble real-life planetary exploration.

We presented, 3DTERRA, which incorporates modifications to several of its original phases, affecting the placement of recharging stops and the computation of the 3-dimensional paths for the UAV and UGV. We tested these modifications with a set of experiments using real data from a DTM of the central uplift of a 30-Km diameter crater in Noachis Terra on Mars, showing that even if the clustering optimization gets more complex and computational performance increases with the 3D environment, 3DTERRA can solve exploratory mission problems with a performance dependent on the battery capacity of the UAVs, the clustering level of the target points in the exploration area, and the complexity of the terrain features.

ACKNOWLEDGMENTS

The work of F. Ropero was co-funded by the European Social Fund and University of Alcalá under the contract number PEJD-2018-PRE/TIC-8176.

Maria D. R-Moreno is supported by the JCLM project SBPLY/19/180501/000024 and the Spanish Ministry of Science and Innovation Project PID2019-109891RB-I00, both under the European Regional Development Fund (FEDER).

REFERENCES

[BMSS05] Wolfram Burgard, Mark Moors, Cyrill Stachniss, and Frank E Schneider. Coordinated multi-robot exploration. *IEEE Transactions on robotics*, 21(3):376–386, 2005.

[Cum] Nigel Cummings. A brief history of the travelling salesman problem. `shorturl.at/pAW59`. Accessed: 1-5-2022.

[Dub57] Lester E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79:497, 1957.

[GD91] David E Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of genetic algorithms*, volume 1, pages 69–93. Elsevier, 1991.

[HNR68] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[MRMC17] Pablo Muñoz, María D R-Moreno, and Bonifacio Castaño. 3Dana: A path planning algorithm for surface robotics. *Engineering Applications of Artificial Intelligence*, 60:175–192, 2017.

[NAS20] NASA. Mars 2020 Mission Perseverance Rover with the Mars Helicopter Scout. `https://mars.nasa.gov/mars2020`, 2020. Accessed: 05-01-2020.

[PS12] Franco P Preparata and Michael I Shamos. *Computational geometry: an introduction*. Springer Science & Business Media, 2012.

[RBTH02] Arthur Richards, John Bellingham, Michael Tillerson, and Jonathan How. Coordination and control of multiple uavs. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 4588, 2002.

[RMRM19] Fernando Ropero, Pablo Muñoz, and María D R-Moreno. TERRA: A path planning algorithm for cooperative ugv–uav exploration. *Engineering Applications of Artificial Intelligence*, 78:260–272, 2019.

[Rop20] Fernando Ropero. *Algorithms for the multi-robot systems on the cooperative exploration and last mile delivery problems.* PhD thesis, Universidad de Alcala, Madrid, Spain, 2020.

[Uni] University of Arizona. HiRISE Digital Terrain Models. Central uplift of a 30-km diameter crater in Noachis Terra. https://www.uahirise.org/hiwish/maps/dtms.jsp.