Rovers: Path Planning and Localization

Andrea Giuseppe Di Francesco, 1836928

Space Robotics System

1 Tasks Overview

In this report, are discussed and illustrated the procedures followed to accomplish 3 different tasks for a mobile robot, namely NASA's *Volatiles Investigating Polar Exploration Rover* (VIPER). These tasks involve the trajectory planning and control of VIPER, together with the development of a localization module. All of the procedures are better explained in the following sections, by highlighting the related difficulties faced during the resolution.

1.1 Input data

The tasks' maps, models a portion of the moon's surface. Such environment presents Permanently Shadowed Regions (PSR) and several areas with steep slopes, these regions are dangerous for the VIPER's safety, and thus must be treated as obstacles to it.
Among the provided data there are:

- Rover's maximum velocity V_{max},
- Rover's axles distance L,
- Maximum traversable slope angle α,
- Initial VIPER pose P_0,
- Intermediate VIPER position P_1,
- Final VIPER position P_f.

As can be inferred by the data, VIPER should accomplish 2 journeys, namely from P_0 to P_1, and subsequently from P_1 to P_f.
In Figure 1, is depicted the mentioned portion of the moon's surface, whereas in Figure 2, are illustrated the position of the previously announced obstacles. Other representations are provided for other specific homework requests.

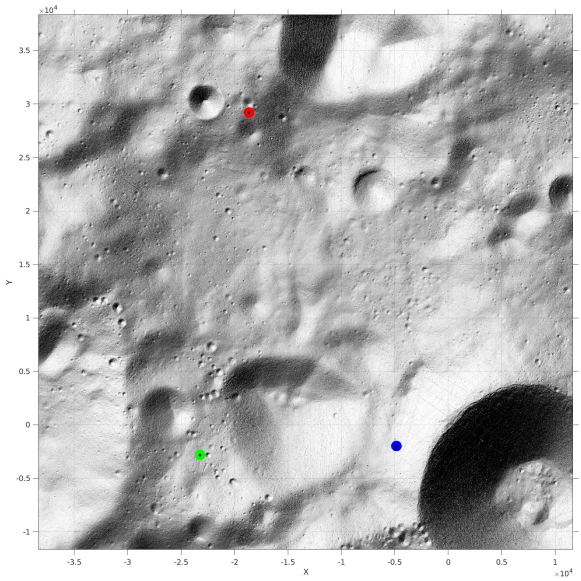

Figure 1: Moon surface. Points on the map represent P0 (green), P1 (blue), Pf (red).

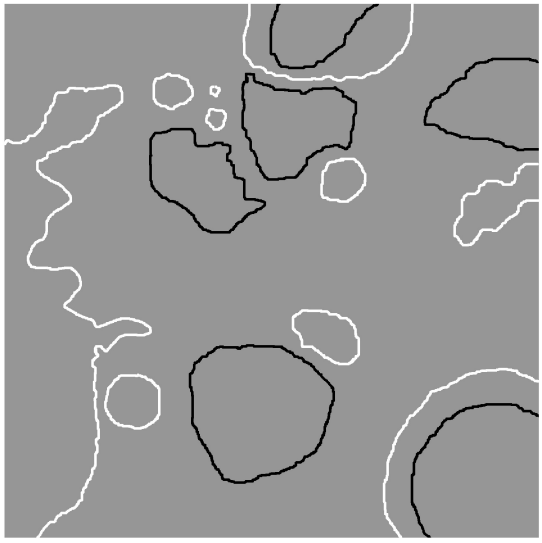

Figure 2: Considering Figure 1, here are depicted the respective obstacles on the moon's surface. PSR (black), Steep slopes (white).

1.2 VIPER kinematic model

Along with the tasks' completion, it is required to update over time the rover's vector of states, and this is done through the commands that are given to it.
A correct update of the VIPER state's vector requires prior knowledge of its kinematic model. Before showing the VIPER kinematic model, it is better to introduce the variables that populate its state's vector q.

$$q = \begin{bmatrix} x & y & \theta \end{bmatrix}^T. \tag{1}$$

Where x and y are simply the cartesian positions of the rover, according to the map reference frame (see Figure 3), and θ is the VIPER's heading angle (see Figure 4).
The kinematic model consists of a set of differential equations that relates the configuration vector q to its derivatives $\dot{q}$. Hereafter the Rover's kinematic model:

$$\begin{aligned} \dot{x} &= v cos(\theta) \\ \dot{y} &= v sin(\theta) \\ \dot{\theta} &= \frac{v}{L} tan(\gamma) \end{aligned} \tag{2}$$

These equations of motion also contain v and γ, which are the commanded velocity and steering angle respectively.
Once the Rover executes an input, it is possible to retrieve its next configuration by integrating the Equations in 2.

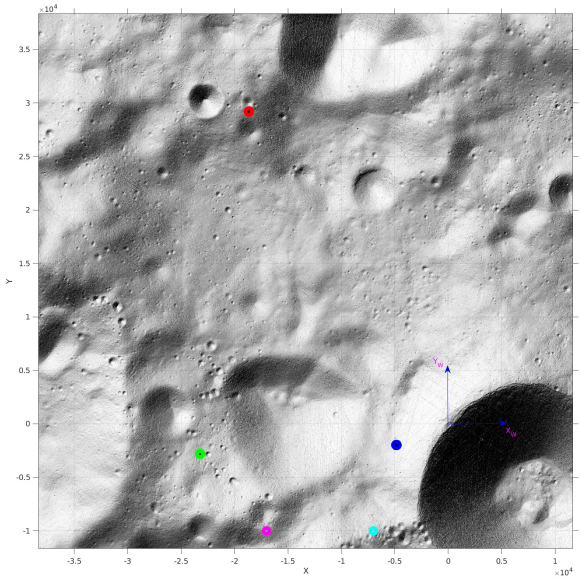

Figure 3: Via points P_{v_1}, and P_{v_2}, the world reference frame in denoted in red.

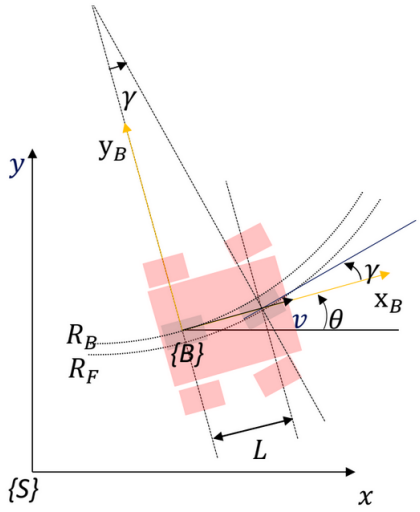

Figure 4: This figure contains a more general representation of the Rover modeling. θ is the heading angle, v and γ are the velocity and steering commands. S is the station frame, whereas B is the base frame.

2 Navigation

2.1 Brief task description

In this task, was asked to compute a trajectory planning from P_0 to P_1. Among the requirements, it is stated that such a trajectory must avoid the PSR.
The current scenario is better described in Figure 3.
After having produced the trajectory (1) it is also requested to compute:

2) The velocity of the rover as a function of time;

3) The heading angle of the rover as a function of time;

4) The rate-of-change of the heading angle as a function of time;

5) The time required to reach the final location.

The results are available at 2.3

2.2 Procedure

To reach P_1, it is not possible to follow a straight line starting from P_0, because of the PSR, thus there were introduced two via points to follow, according to Figure 3, before going to the final location. Here are reported numerical data

of the initial, final, and via points (P_{v_i}, $i \in \{1, 2\}$).

$$\begin{aligned} P_0 &= \begin{bmatrix} -23225 & -2815 & \theta_{P_0} \end{bmatrix}^T \\ P_0 &= \begin{bmatrix} -4855 & -1975 & \theta_{P_1} \end{bmatrix}^T \\ P_{v_1} &= \begin{bmatrix} -17000 & -10000 & \theta_{P_{v_1}} \end{bmatrix}^T \\ P_{v_2} &= \begin{bmatrix} -7000 & -10000 & \theta_{P_{v_2}} \end{bmatrix}^T \end{aligned} \tag{3}$$

These Equations follow the same structure of Equation 1.
After having established the via points, must be computed the real trajectory as a function of time.
To do so, it was applied the trajectory control strategy named 'Moving to a point', which consists of a control law that updates the Rover's state proportionally to the current error, at each timestep. In these experiments, the chosen timestep was 1 Hz. There were also conducted experiments at 10 Hz, but these were not taken into consideration for the following tasks.
The Moving to a point command definition is described as follows.

$$\begin{aligned} v^* &= K_v \cdot \sqrt{(x^* - x)^2 + (y^* - y)^2} \\ \gamma^* &= K_h \cdot (\theta^* - \theta) \end{aligned} \tag{4}$$

Where $\cdot^*$ are the desired values, computed according to the current target, and in particular $\theta^* = atan2(y^* - y, x^* - x)$.
As regards the tuning of K_v and K_h, in general, they can be chosen arbitrarily, but in our case, we have a constraint on v^*, such that (s.t.):

$$v^* \leq V_{max} \tag{5}$$

Due to the constraint, K_v was determined at each timestep as follows.

$$\begin{aligned} K_v &= \frac{V_{max} - \epsilon}{error}, \\ error &= \sqrt{(x^* - x)^2 + (y^* - y)^2} \end{aligned} \tag{6}$$

Where ϵ is 0.1, and it was chosen just to not fully saturate the Rover motors.
Since the configuration K_v was computed according to Equation 6, the velocity constraint was never violated, and as the error decreased the K_v increased.

2.3 Results

The 'Moving to a point' algorithm was stopped when the Rover was within 1 meter in the goal range.
In Figure 5 is depicted the trajectory of the Rover according to the exercise's requests (1).

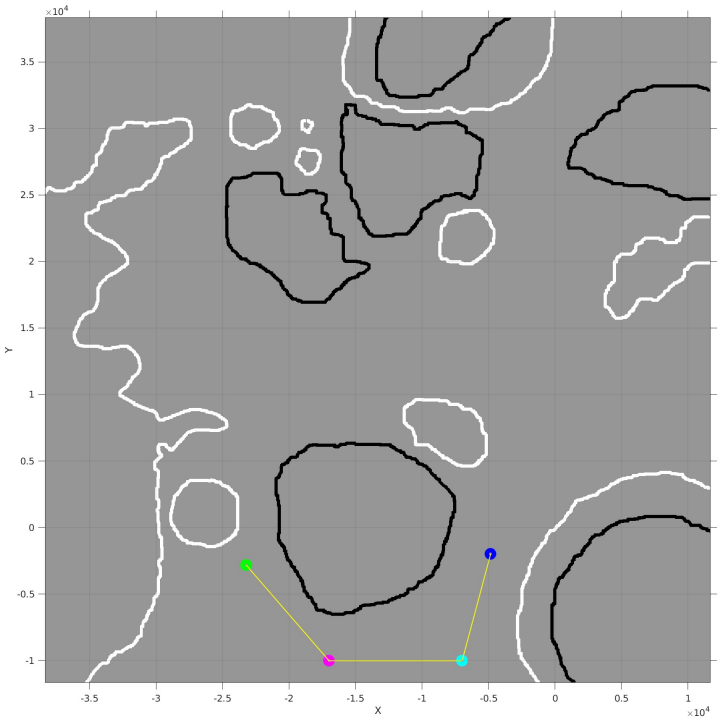

Figure 5: Rover trajectory across P_0, P_{v_1}, P_{v_2}, P_1.

2). **Velocity of the rover as a function of time**
In Figure 6, there is the trend of the Rover's velocity over time, it is clear that it stays constant, since the commanded v^* it is always computed as explained in Equation 6.

3). and 4). **The heading angle of the rover and its rate, as a function of time**
In Figures 7, 8 are displayed the trends of the heading angle and its derivative. It is noticeable that whenever the goal destination changes the heading angle rate increases immediately, and then goes back to 0 when the heading angle desired θ^* is finally reached.

5). **The time required to reach the final location.**
The required time to reach the final location is computed by converting the seconds into hours. Since the total journey has lasted 139,811 seconds, which corresponds to 38.8364 hours. The longer trajectory was the last one, namely from P_{v_2} to P_1.

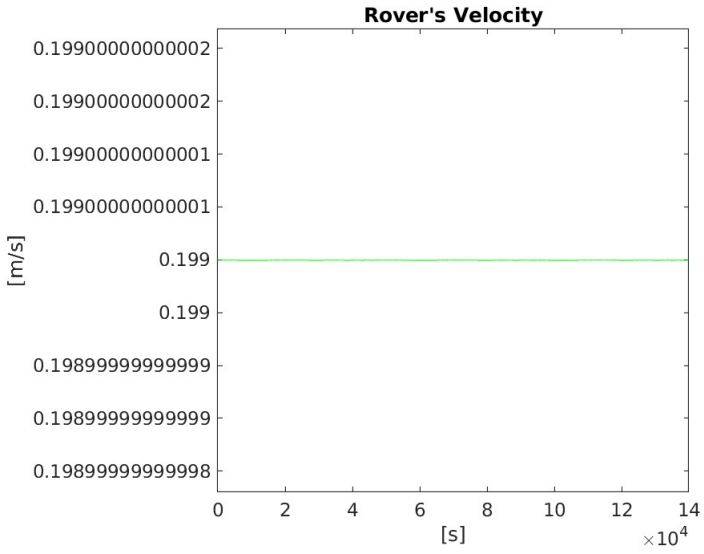

Figure 6: Rover's velocity during the execution of the trajectory in the first task.

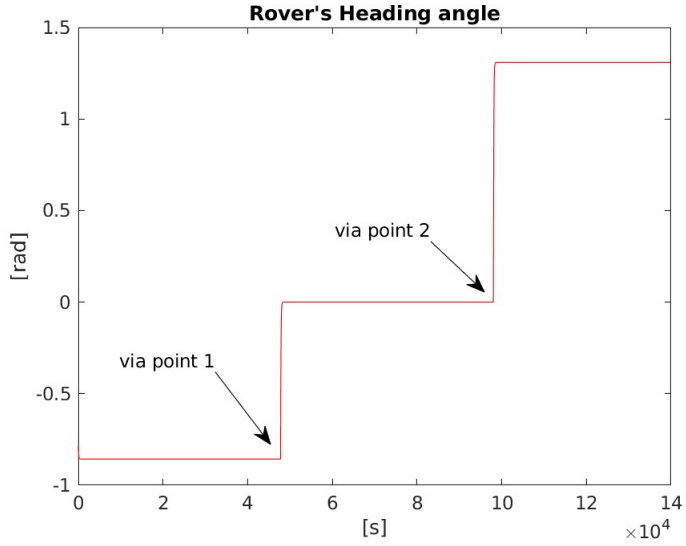

Figure 7: Rover's heading angle during the execution of the trajectory in the first task.

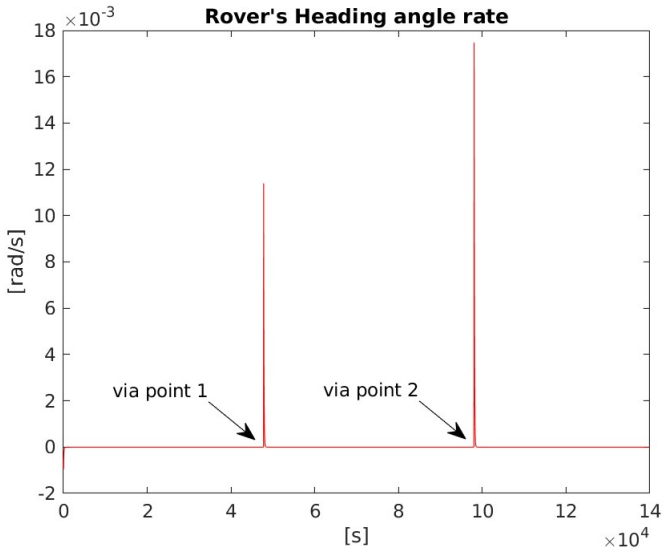

Figure 8: Rover's heading angle rate during the execution of the trajectory in the first task.

3 Path Planning

3.1 Brief task description

Path planning task involves the computation of the minimum length path from the goal destination of the first journey, namely P_1, up to P_f. In this case, it is required to compute the minimum path distance through the A^* algorithm (see Figure 9), where the PSR and the steep slopes regions in Figure 2 are obstacles that cannot be crossed by the Rover. Results are available in Section 3.4.

3.2 Mandatory task

The A^* algorithm requires a $G(\cdot)$ and a $h(\cdot)$ function, s.t. $G(\cdot)$ is a measure of how far from the initial position, the Rover is located, while $h(\cdot)$ returns an estimate of the goal location distance. In order to make the A^* work correctly and find the optimal path, the heuristic must not overestimate the goal distance, for this reason, it is fundamental to use an admissible heuristic. Given that our Rover is moving in an 8-way grid, an admissible heuristic for this setting is the following one.

$$h(n) = max(dx, dy) + (\sqrt{2} - 1)(dx, dy). \tag{7}$$

Where n is a generic node, and $dx = |x(n) - x_f|$, $dy = |y(n) - y_f|$.
As regards $G(n)$ it is computed considering each map's pixel with a cost of 5, when moving straight, and equals to $5\sqrt{2}$ when moving diagonally.

Algorithm 0.0.1 A* Search Algorithm

```
function A*(n_start, n_goal)
    openSet := {n_start}
    cameFromSet := ∅
    G := ∞
    F := ∞
    G(n_start) := 0
    F(n_start) := h(n_start)
    while openSet is not empty
        n_curr := node in openSet with lowest F value
        if n_curr = n_goal then
            return PATH RECONSTRUCTION(n_goal, cameFromSet)
        Remove n_curr from openSet
        for each node n neighbor of n_curr do
            c(n, n_curr) := cost of the edge (n, n_curr)
            G_tentative = G(n_curr) + c(n, n_curr)
            if G_tentative ≤ G(n) then
                G(n) := G_tentative
                F(n) = G(n) + h(n)
                cameFromSet(n) := n_curr
                if n ∉ openSet then
                    Add n to openSet
```

Figure 9: Pseudo-code of the A-Star algorithm. 'PathReconstruction(·,·)' is an additional function that returns the sequence of nodes for the minimum length path, once the path to the goal node has been found.

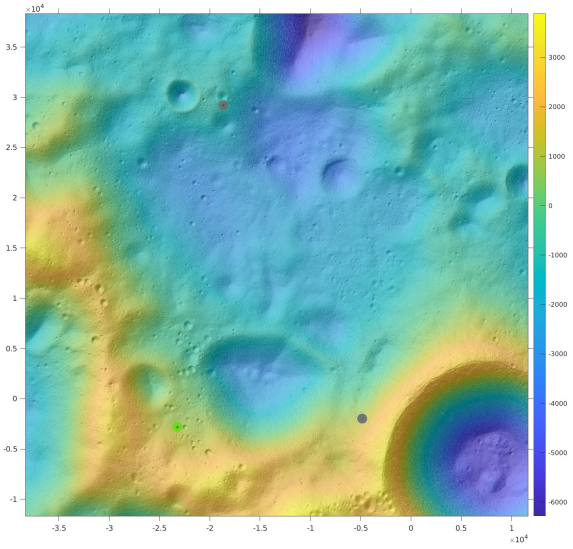

Figure 10: DEM of the moon surface.

3.3 Optional task

Together with the standard obstacle map (see Figure 2), it was also provided a *digital elevation map* (DEM) of the environment (see Figure 10). As an optional task, it was requested to repeat the same experiment as the mandatory one, but now it was requested to consider the elevation as an additional constraint.
In fact in Section 1, it was defined as a variable α, which is the maximum slope that the Rover can cross. To conduct the optional task, at each node, the node's neighbors were evaluated according to α, for instance, if the angle obtained between the inclined and the planar distance of the nodes is higher than α, that transition must not be allowed. Let us consider a pair of nodes (n_{curr}, n_{neigh}) the comparison with α is computed as follows.

$$atan2(elevation(n_{curr}, n_{neigh}), cost(n_{curr}, n_{neigh})) \leq \alpha \tag{8}$$

Where n_{curr} is the current node, and n_{neigh} is the candidate neighbor in which the motion is directed.
If the motion is allowed, then the total transition cost (from n_{curr} to n_{neigh}) is computed as the value of the triangle's hypotenuse that has for cathexes the $elevation(n_{curr}, n_{neigh})$ and $cost(n_{curr}, n_{neigh})$.

3.4 Results

In this task, was requested to compute:

1. The length of the planned path;
2. The planned path.

These should be computed also for the optional task.
1).**MANDATORY**: The length of the found optimal path was 39766 meters.
2).**MANDATORY**: In Figure 11 is depicted the optimal path followed by the rover, and in Figure 12 is illustrated a portion of the expanded nodes, that were needed to find the optimal one.
Unfortunately, the path in Figure 11 was computed considering the line 16 in the Algorithm 9 as $G_{tentative} < G(n)$, instead of using $\leq$. This means that each node n was added to the *OpenSet* whenever its $G_{tentative}$ value was lower than the current $G(\cdot)$, it was not considered the equality case to satisfy the condition. This probably has increased the optimum length found by the A^*.
The optional task presented additional difficulties since it was necessary to include the slope constraint. In the end, no solution was found, but it does not mean that the solution did not exist. In fact, some additional limitations were introduced due to the computational memory limits.
The limitations comprised the acceptance of nodes to be analyzed, if and only if they had a heuristic value higher than a threshold. Formally the rule to accept a node n in the *OpenSet* was the following one.

$$h(n) \leq h(P_1). \tag{9}$$

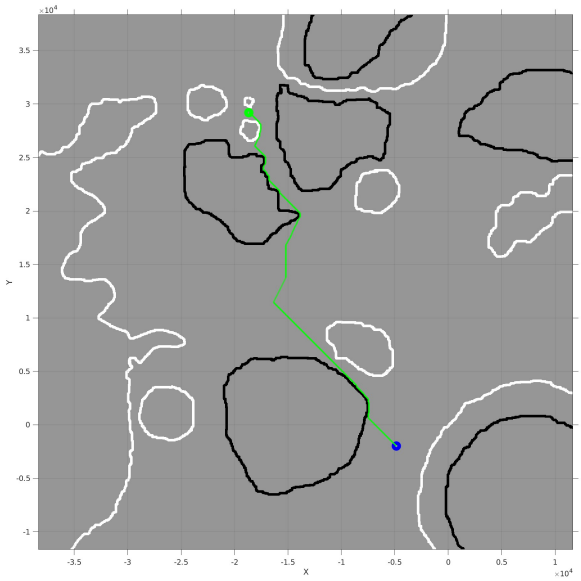

Figure 11: A^* found optimal path.

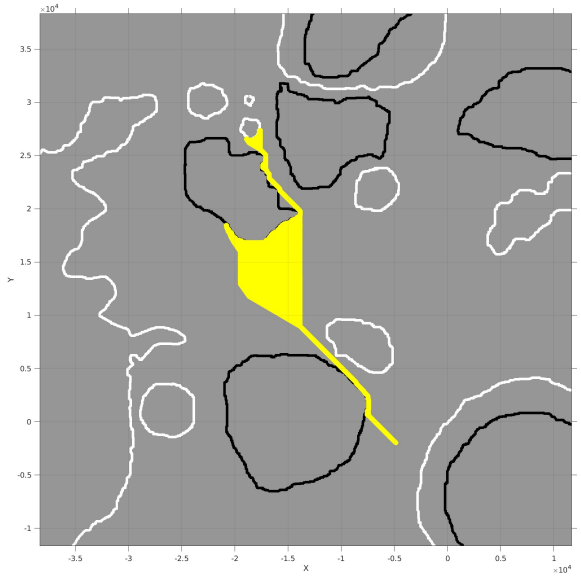

Figure 12: Expanded nodes, with the A^* algorithm.

Where P_1, was the initial node.
Another limitation was on the x-axis values to be explored, namely, the nodes were considered as obstacles if $x_W \notin \{-1000, -20000\}$.

Searching for a path with these limitations was not possible. In Figure 13 is displayed the nodes' expansion before the algorithm stopped.

A potential solution could take into account a different method to compute

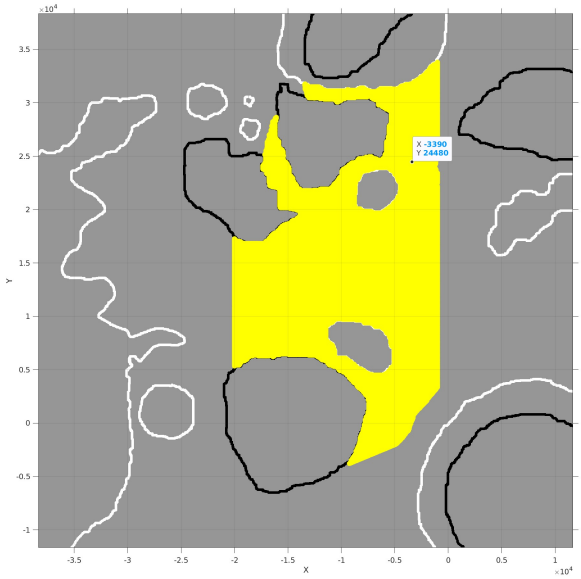

Figure 13: A^* expansion, considering the limitation on the slope. It is also possible to observe the additional limitations included to reduce the computational space.

the elevation between two nodes.
It would be possible to consider it as the average of an arbitrary number of neighbors (that belong to the same direction) in the elevation map. This strategy is typically used in geodesy for the elevation's estimate.

4 Rover Localization

4.1 Brief task description

In the Localization task, it was requested to consider the measures obtained in Task 1 (see Section 2), and perform a path reconstruction of the final trajectory. Reconstructing the path, through past measures, consists of using the collected odometry parameters over time, in the already performed trajectory. In the second part of the task, the odometric measures are also mixed with external measures, retrieved through a LIDAR sensor. The LIDAR can provide the range and bearing angle relative to some known landmarks, on the moon's surface. This last part is expected to achieve an improved path reconstruction w.r.t. the previous, thanks to the adoption of the *Extended Kalman Filter* (EKF).

To carry out the computations, there were provided additional data, among which there are the odometry and the LIDAR uncertainties.
As in Task 1, the odometric and LIDAR measurements are acquired at 1 Hz. To complete the task must be retrieved the following results:

1. The trajectory of the rover from P0 to P1 reconstructed through the dead reckoning and the ellipses of uncertainties along the path;
2. The trajectory of the rover from P0 to P1 reconstructed through the combination of the odometer and the LIDAR measurements and the ellipses of uncertainties along the path;
3. The evolution of the square root of the determinant of the covariance matrix in both cases.

The results are in Section 4.3.

4.2 Procedure

4.2.1 Odometry: Dead Reckoning

The first part of the localization task is focused on path reconstruction, with the aid of only odometry. Thus, it is in our interest to get a sequence of configurations, which resembles the desired path and that is computed according to the task 1 quantities. The reconstructed path update rule is the one that follows.

$$\begin{aligned} \hat{x}_{k+1} &= \hat{x}_k + (\delta_{d_k} + v_d)cos(\hat{\theta_k}) \\ \hat{y}_{k+1} &= \hat{y}_k + (\delta_{d_k} + v_d)sin(\hat{\theta_k}) \\ \hat{\theta}_{k+1} &= \hat{\theta}_k + \delta_{\theta_k} + v_\theta \end{aligned} \tag{10}$$

$$\hat{P}_{k+1} = F_{q_k}\hat{P}_k F_{q_k}^T + F_{v_k}\hat{V}F_{v_k}^T \tag{11}$$

In Equation 10, is performed a localization, through the dead reckoning strategy. This means, that there is no external feedback, but the update is computed solely on internal measurements.
In this setting, the reconstruction error is dependent on the discretization error (in task 1 the kinematic model was integrated), and also on the noise $v_k = \begin{bmatrix} v_{d_k} & v_{\theta_k} \end{bmatrix}^T$. As regards the other quantities in Equation 10, such as δ_{d_k}, and δ_{θ_k}, these are the odometric measures, computed from the gold trajectory $q_{gold,k}$, namely the one achieved in task 1. They do represent the travelled distance, from $q_{gold,k}$ to $q_{gold,k+1}$, and the heading angle variation from $q_{gold,k}$ to $q_{gold,k+1}$. Given that:

$$q_{gold,k} = \begin{bmatrix} x_{gold,k} & y_{gold,k} & \theta_{gold,k} \end{bmatrix}^T, \tag{12}$$

The odometric quantities are computed as follows.

$$\begin{aligned} \delta_{d_k} &= \sqrt{(x_{gold,k+1} - x_{gold,k})^2 + (y_{gold,k+1} - y_{gold,k})^2} \\ \delta_{\theta_k} &= \theta_{gold,k+1} - \theta_{gold,k} \end{aligned} \tag{13}$$

These quantities are available during the reconstruction phase because they were recorded by only internal sensors, so after the configuration update we get some error, that is amplified over the path reconstruction. In Equation 11, the covariance matrix estimate is defined, it gives information on how much the system is certain about its current estimate. In a dead reckoning scenario, it is expected to record an increasing trend for the determinant of P over time, since the system can easily accumulate errors. The other quantities in the Equation 11 refer to the jacobian matrices for the set of equations 10, w.r.t q_k (F_{q_k}), and w.r.t. the noise v_k (F_{v_k}).

4.2.2 Odometry + EKF

To fully employ the Kalman engine, it is required to compute at each estimate of $\hat{q}_{k+1}$, and $\hat{P}_{k+1}$, an additional update of these, that now takes into account also external sensor's measurements. In our case, a LIDAR sensor is available, and also several landmarks can be found on the map. This situation is depicted in Figure 14.

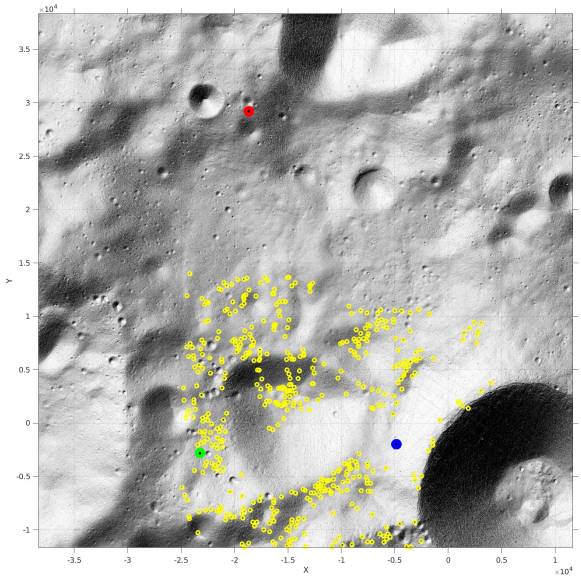

Figure 14: Landmarks position on the map.

The intuition behind the Kalman filter is that the external observations of the landmarks can adjust the path reconstruction, and then reduce the uncertainty on the state estimate. This is possible whenever the landmarks in the current configuration are available, if this is not valid then, only odometry is exploited for the estimate. Once observations are provided the following Equations can

be computed.

$$\begin{aligned} p_i &= \begin{bmatrix} x_i & y_i \end{bmatrix}^T, \\ h(q, p_i) &= \begin{bmatrix} \sqrt{(y_i - y)^2 + (x_i - x)^2} \\ atan2(y_i - y, x_i - x) - \theta \end{bmatrix}, \\ z_{k+1} &= h(q_{gold,k+1}, p_i) + \begin{bmatrix} w_r \\ w_\beta \end{bmatrix}, \\ \nu_{k+1} &= z_{k+1} - h(\hat{q}_{k+1}, p_i) \end{aligned} \tag{14}$$

p_i is the landmark position in the world frame W. At each iteration, i have chosen to consider **only one landmark**, the one which was the **closest** to the VIPER's configuration. $h_1(q, p_i)$ computes the range from the current configuration to the landmark i, and $h_2(q, p_i)$ is the correspondant bearing angle. z_{k+1} represents the LIDAR measure, it is computed as $h(\cdot, p_i)$, computed in the gold path, and additionally is considered the sensor noise.
Finally can be computed the *innovation* ν_{k+1} as the discrepancy between the external measure's estimate, and the actual external measure.
The last step is the computation of the Kalman gain K_{k+1}, which is performed as follows.

$$K_{k+1} = \hat{P}_{k+1} H_{\hat{q}_{k+1}}^T (H_{\hat{q}_{k+1}} \hat{P}_{k+1} H_{\hat{q}_{k+1}}^T + H_W W H_W^T)^{-1} \tag{15}$$

$H_{\hat{q}_{k+1}}$ is the jacobian of the observations w.r.t. q, and H_W w.r.t. $w = \begin{bmatrix} w_r & w_\beta \end{bmatrix}^T$. W is the covariance matrix of the LIDAR's noise. The definitions of these quantities are available in the scripts.
The q, and P estimates follow:

$$\begin{aligned} q_{k+1} &= \hat{q}_{k+1} + K_{k+1} \nu_{k+1} \\ P_{k+1} &= (I - K_{k+1} H_{q_{k+1}}) \hat{P}_{k+1} \end{aligned} \tag{16}$$

q_{k+1}, and P_{k+1}, define at each step the reconstructed path, with the uncertainties. At this stage, it is expected that the path should be more similar to the original, and the uncertainties should not be much relevant.

4.3 Results

1) **The trajectory of the rover from P0 to P1 reconstructed through the dead reckoning and the ellipses of uncertainties along the path**.
In Figure 15 is shown the reconstructed path. As expected the path reconstruction is ill-conditioned, due to the discretization+noise errors.
One of the main issues caused by the dead reckoning is that the ellipses of uncertainties can fall into obstacles, that should not be crossed. From the Figure is clear that the Rover could end up in one obstacle. Anyway when sensors are not available, one can try to find a set of via points that decrease such risk. For example, in previous experiments the via points set was the one in Figure 16, and the dead reckoning result was the one depicted in Figure 17.

Avoiding the obstacles in an uncertain scenario could be fostered by a better

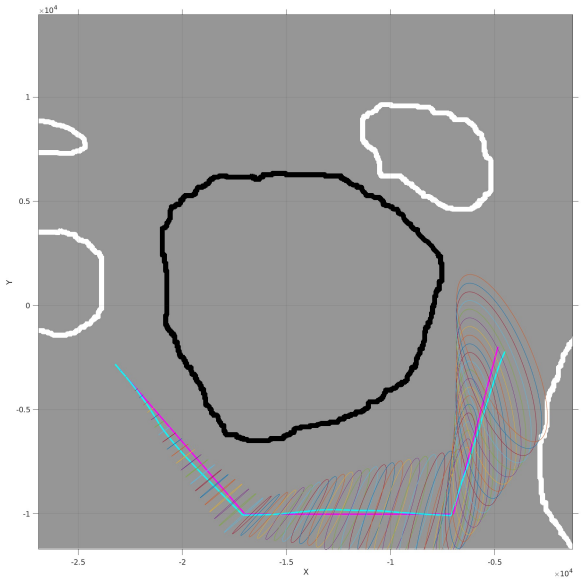

Figure 15: Path reconstruction with dead reckoning. MAGENTA: Desired path, CYAN: Reconstructed path

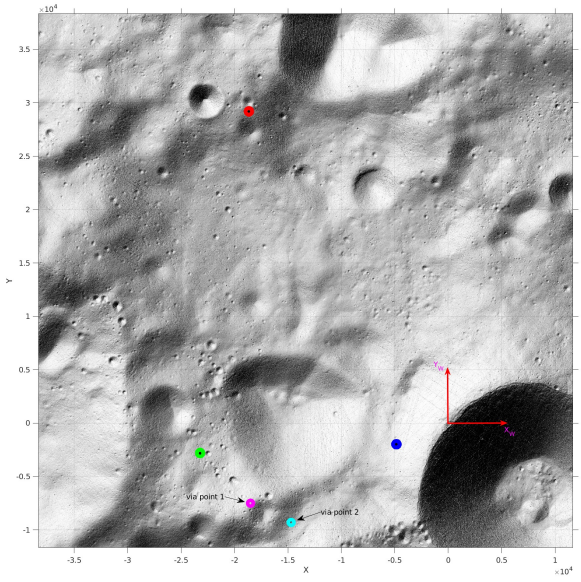

Figure 16: Old via points set, which has not performed well in the last task, especially when computing the ellipses of uncertainties.

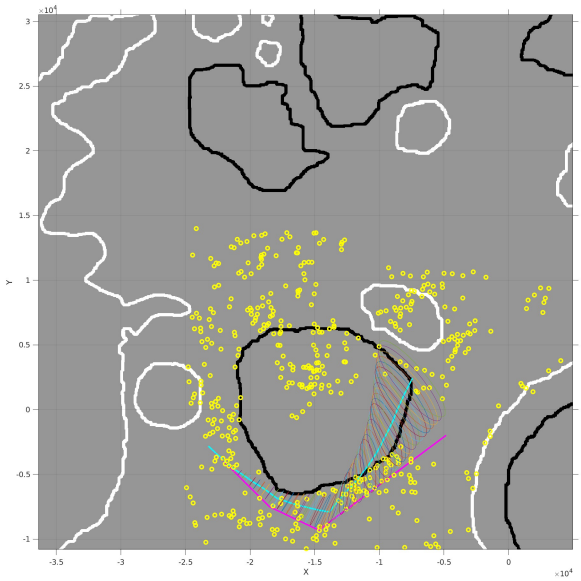

Figure 17: In this Figure is shown a dead reckoning result, where the current via points set has provided a dangerous trend of the ellipses' dimension.

via points' choice.

2) **The trajectory of the rover from P0 to P1 reconstructed through the combination of the odometer and the LIDAR measurements and the ellipses of uncertainties along the path.**

The path reconstruction with the EKF is shown in Figure 18. It is hard to see the displacement between the desired, and reconstructed path since the EKF managed to reconstruct it properly. In Figure 19, there is a better view of the committed error during the reconstruction.

3) **The evolution of the square root of the determinant of the covariance matrix in both cases.**

In Figures 20, 21, are displayed the trends of the squared root of the P matrix's determinants respectively. It is clear that in the EKF setting, the state's uncertainty converges to zero, which is not a surprise since the noises are modeled as white noise (zero mean gaussian), and this algorithm is effective with this type of choice.

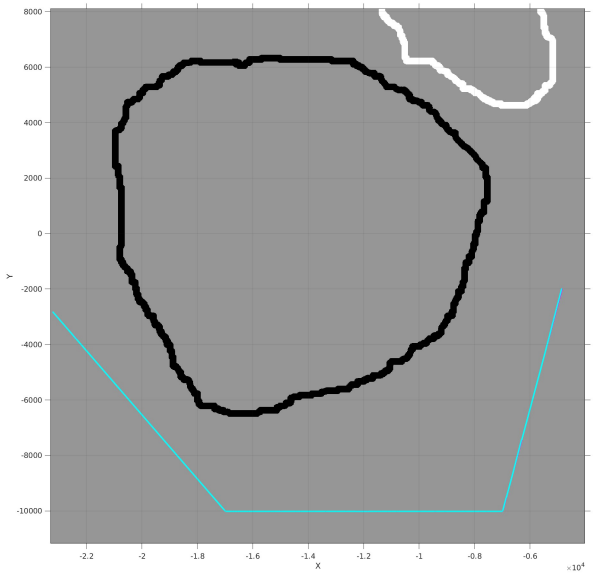

Figure 18: Path reconstruction with Odometry + EKF. MAGENTA: Desired path, CYAN: Reconstructed path.

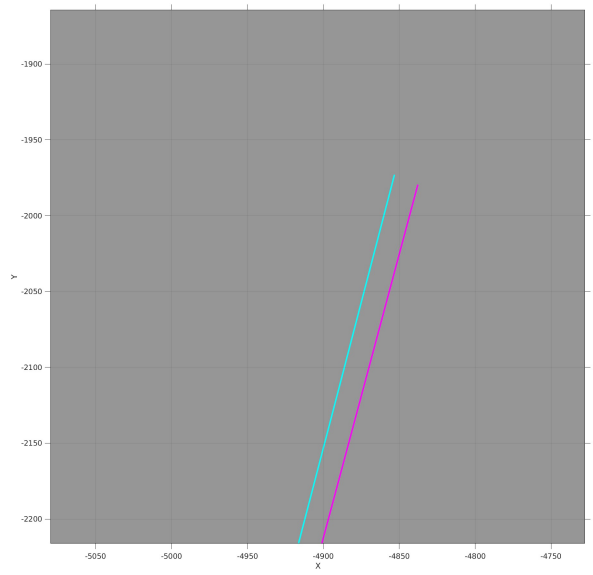

Figure 19: Closer view of the path reconstruction with the EKF.

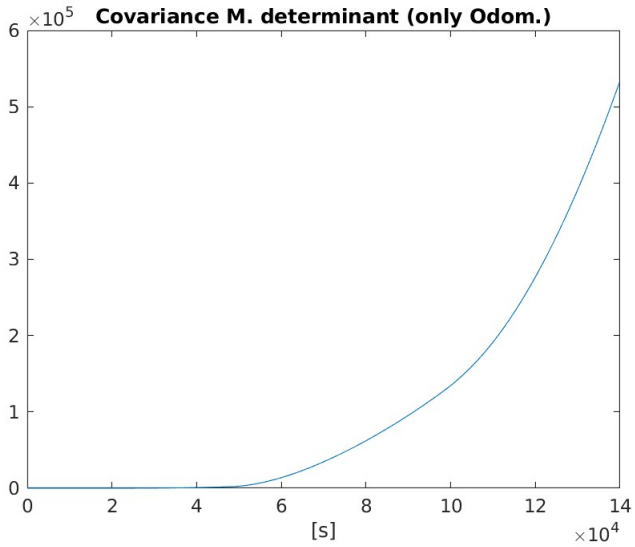

Figure 20: Squared root of the P matrix's determinant in the dead reckoning setting.

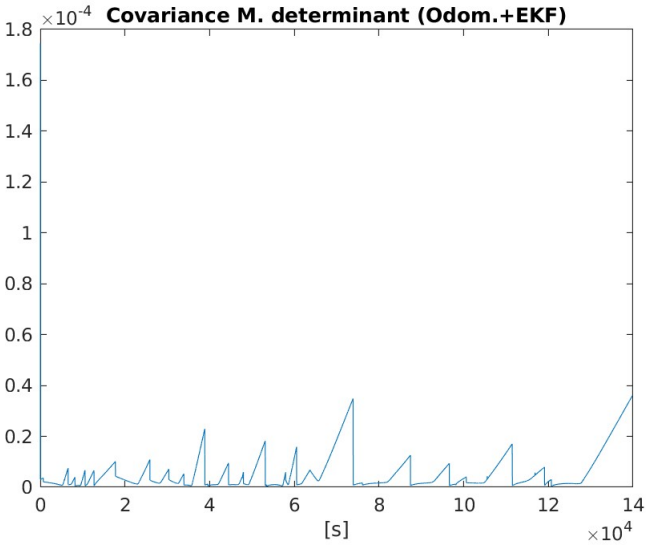

Figure 21: Squared root of the P matrix's determinant in the EKF setting.