Efficient Autonomous Navigation for Planetary Rovers with Limited Resources

Levin Gerdes*
European Space Agency
Keplerlaan 1, 2200 AG Noordwijk, NL
levin.gerdes@esa.int

Martin Azkarate*
European Space Agency
Keplerlaan 1, 2200 AG Noordwijk, NL
martin.azkarate@esa.int

J. Ricardo Sánchez-Ibáñez
Department of Systems Engineering
and Automation, Universidad de Málaga
C/Ortiz Ramos s/n, 29071 Málaga, Spain
ricardosan@uma.es

Luc Joudrier
European Space Agency
Keplerlaan 1, 2200 AG Noordwijk, NL
luc.joudrier@esa.int

Carlos Perez Del Pulgar
Department of Systems Engineering and Automation
Universidad de Málaga
C/Ortiz Ramos s/n, 29071 Málaga, Spain
carlosperez@uma.es

Abstract

Rovers operating on Mars are in need of more and more autonomous features to fulfill their challenging mission requirements. However, the inherent constraints of space systems make the implementation of complex algorithms an expensive and difficult task. In this paper we propose a control architecture for autonomous navigation. Efficient implementations of autonomous features are built on top of the current ExoMars navigation method, enhancing the safety and traversing capabilities of the rover. These features allow the rover to detect and avoid hazards and perform long traverses by following a roughly safe path planned by operators on ground. The control architecture implementing the proposed navigation mode has been tested during a field test campaign on a planetary analogue terrain. The experiments evaluated the proposed approach, autonomously completing two long traverses while avoiding hazards. The approach only relies on the optical Localization Cameras stereobench, a sensor that is found in all rovers launched so far, and potentially allows for computationally inexpensive long-range autonomous navigation in terrains of medium difficulty.

1 Introduction

In 2020, the European Space Agency (ESA) is going to launch the ExoMars rover, one of its most prestigious missions yet. ExoMars is a six-wheeled rover that embarks a complex drill system and laboratory equipment to sample the subsurface soil of Mars in search for traces of life. The Trace Gas Orbiter (TGO), which assumed its orbit in early 2018, will act as a communication relay and enable bi-directional communication

*Corresponding authors

between rover and ground. This communication, though, may only happen twice per Sol (Martian day) due to constraints on the allocation of deep space antennas and the TGO's orbital trajectory, that shall only have visibility of the rover at certain passes. Only during these communication windows may the rover send telemetry data back to Earth and receive new commands from operators on ground for the upcoming Sol. These communication constraints, in addition to the signal propagation delay between Earth and Mars, render continuous (online), direct, and supervised control impossible. It is therefore relevant to address the autonomous capabilities of the rover, such that they could potentially increase the scientific return of the overall mission. In particular, we focus here on the navigation aspects of the rover autonomy.

There exist many examples of autonomous navigation on Earthly applications. Since the DARPA Grand and Urban Challenges, humans have shown their capability to build and run autonomously driven vehicles. Nowadays, many car manufacturers and companies are developing their own self-driving cars capable of safely transporting humans in traffic areas. However, we seldom find examples of autonomously navigating rovers in space. Only in the latest years, Mars Exploration Rovers (MER) and Mars Science Laboratory (MSL)/Curiosity rover have exploited, under certain circumstances, autonomous navigation capabilities and features (Maimone, 2017). The constraints that the space environment and space systems impose usually make the solutions adopted in Earth applications unfeasible. To start with, the already mentioned remoteness implies that the rovers are completely isolated and alone with no available connection to any data servers or Global Navigation Satellite System (GNSS). Plus, they are roving on unknown, difficult and unstructured surfaces such as the one shown in Figure 1, with little *a priori* knowledge of the terrain based on orbiter imagery. While these are useful to get a first impression of potential landing sites, they cannot deliver the same level of detail as the maps of cities, lands, and roads (Jet Propulsion Lab, 2018; McEwen et al., 2007). As a consequence, there may be obstacles which are at the same time too small to be captured by the orbiter and too large for the rover to safely overcome. Moreover, the harsh environmental conditions, such as vacuum, extreme temperatures, and radiation have created a permanent technology gap for the available components for space. The latest space qualified computers are only as capable as a 10-year-old computer on Earth (Lentaris et al., 2018). Sensors such as scanning LiDARs, located in all the self-driving cars on Earth, have not yet been qualified for space, as their spinning mirrors would break during their launch to space, not being capable to withstand the structural stress. In addition, limitations in resources like mass and power play a major role in the design of the system and component selection that need to be highly optimized. And finally, the criticality and unique opportunities that space systems bring make their development to adhere to strong Reliability Availability Maintainability and Safety (RAMS) requirements. In particular, all Flight SW code is thoroughly tested against RAMS and many of the commonly used C++ libraries that run on Earth applications do not qualify. All these constraints render the implementations of autonomous navigation features and Computer Vision (CV) algorithms for space a very challenging task, where computational power and efficiency are two of the most important design drivers.

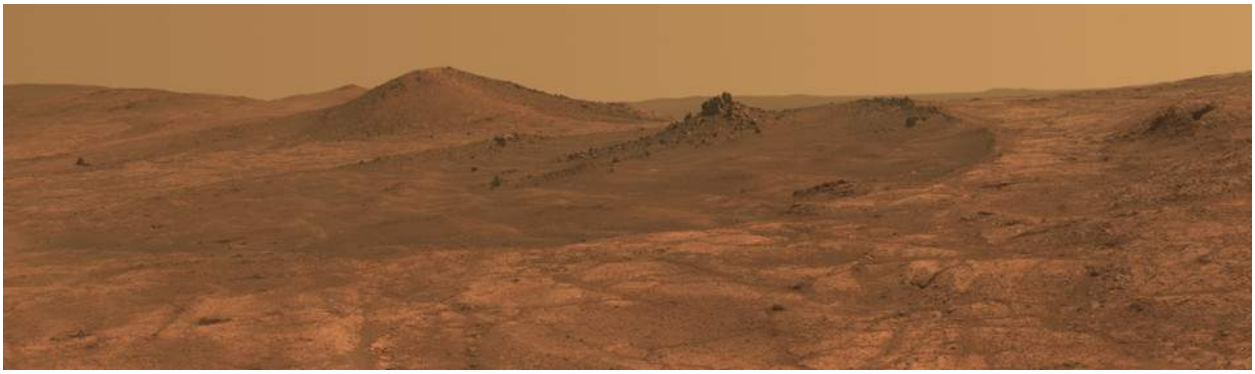


Figure 1: A panorama taken on Mars (credit NASA).

Work in (Matthies et al., 2007; Maimone et al., 2007; Cheng et al., 2005) shows the first CV algorithms used by NASA's MER in order to achieve on-board relative localization using Visual Odometry (VO), and (Maimone, 2017) gives more details on the latest features for autonomous navigation run on Mars rovers.

In Europe, two main implementations of VO for space exist, one developed by Oxford University (Shaw et al., 2013) and one developed under the ESA project SPARTAN (Kostavelis et al., 2011). Successive initiatives of SPARTAN made significant effort towards optimizing and efficiently bringing localization and mapping processes to space representative FPGA hardware implementations. With the aim of increasing the Technology Readiness Level (TRL) of these technologies few ExoMars-like field test campaigns have been run in the Atacama Desert. In the SEEKER campaign (Woods et al., 2014) the rover managed to autonomously traverse a total of 5.05 km during one day. In addition, the French National Centre for Space Studies (CNES) has also developed their solution for AutoNav (Moreno, 2013; Bousquet, 2011) which they have proved to work in both simulation and shorter field experiments. All in all, while ESA and Europe have been working for years on solutions for fully autonomous navigation capabilities, as of the moment of writing, these have not been implemented for the ExoMars rover yet, as only a VO process in running onboard. Apart from project constraints, main limitations for not having these implemented yet are also due to efficiency and computational resource issues. Some approaches for an autonomously navigating rover (Correal and Pajares, 2011) proposed the strategy to stop every few meters to remap its surroundings and evaluate traversability. The low computational on-board capabilities would need tens of seconds to produce a map, and in the order of seconds to run a self-localization iteration. For this reason both the localization and mapping processes would need to run alternating, with localization running while driving and mapping while stopped.

Current plans foresee that the ExoMars rover will be operated in a similar fashion as the early days of NASA's Curiosity rover, i.e., for each Sol, it will receive a detailed list of tasks authored by a team of specialists. The tasks shall include the execution of a path which is planned on ground in the form of navigation waypoints that the rover will need to follow. The navigating capabilities of the rover are limited to trajectory control (i.e., path following) and on board localization. The rover operators rely on the rover's sensor data in order to gauge the terrain's traversability and estimate risks. The main sensors used for navigation and localization are stereo cameras. Taking into account potential occlusions and decreasing stereo quality over distance, the longer the traverse the more difficult it is to guarantee a safe navigation. Due to the inherent need for conservative planning, the commanded distance to travel per Sol may be shorter than what the rover's resources could potentially achieve in terms of energy required for locomotion, making this a limiting factor to the potential science return.

In this paper, we propose the use of computationally efficient safety features in order to compensate for unavailable or computationally expensive autonomous navigation. For the ExoMars case in particular, these features could allow for a less conservative planning while still ensuring the safety of the rover. One of the features is automatic hazard detection. Counting on a rover which can reliably detect hazards and stop automatically, the operators are free to command longer traverses knowing that the rover will in any case be safe. As a second feature, we investigate how we can increase the level of autonomy by allowing dynamic on-board re-planning for hazard avoidance. Instead of just stopping in front of a hazard, we want to register the obstacle, determine its position with respect to the rover and circumvent the obstacle in order to automatically get back on track.

It is important to note that these features come at a low computational overhead given that the algorithms to compute stereo disparities, on which we base the hazard detector, are already being applied to parts of the camera's image frames for localization. As for the avoidance feature, while this would require expanding the current ExoMars on-board capabilities, it would only be needed to run at discrete events and therefore it is expected that it could have little impact on the overall power consumption. We emphasize efficiency as an important and motivational aspect of our work, as can be deemed from the aforementioned resource limitations and constraints of space missions.

Rovers have to navigate unexplored and potentially difficult terrain on other planets. The focus of our work is in the use-case of Mars exploration. Since a rover is essentially the vessel for a science mission, it has to maximize its scientific return by being able to navigate efficiently, especially considering the limited lifetime they are designed for. While the approach presented here is of interest for the ExoMars mission case, these features may become more relevant in future Mars exploration mission such as the Mars Sample Return

where the travel distance requirements are significantly larger (Vago et al., 2017; Merlo et al., 2013; Wilson, 2013).

The rest of the paper is structured as follows. In section 2 the different approaches to navigation are explained. Later on, in section 3, the main building blocks needed to implement our proposed strategy for autonomous navigation are described. Then, in section 4 the experimental results of running the implemented strategy on a field test campaign are detailed. Finally, in section 5 we summarize the conclusions that can be drawn from these results.

2 Navigation Methodologies

There are multiple possible strategies for operating and navigating a rover in Mars. These are mainly dependant on the on board capabilities that may allow for more or less autonomous operation of the system. In this section we discuss four general approaches, ranging from fully pre-planned paths, also referred to as "blind path following", to fully autonomous navigation. We assume an autonomously navigating rover to be one with the capabilities of: assessing the surrounding risks, planning a safe path to the commanded navigation target and being able to follow this planned path. These capabilities would usually require several software modules running in parallel, and in particular, processes to compute the rover localization and environment mapping are needed. As later is explain, is in between the path following and fully autonomous navigation approaches that we position our proposed solutions. Therefore, they increase the level of autonomy compared to the first approach but do not (try to) reach full autonomy, as they do not actually implement a continuous mapping process.

2.1 Fully Pre-Planned Paths

The current approach to the ExoMars rover operations is to have a team of experts plan a detailed sequence of tasks for the rover to execute on the following Sol, aslo referred to as *Tactical planning*. When a task involves navigation, the command includes a list of navigation coordinates, slightly separated from each other (few centimeters), that the rover must follow. These coordinates must form a safely traversable path as the rover will not be able to detect any potential obstacles along the path. For this approach, the team of operators relies on the rover's sensor input received from the previous Sol(s), as well as the low-resolution imagery from Mars orbiters. Relevant telemetry data may include housekeeping data of different subsystems, camera sensor images and the products generated from these such as stitched panoramas and short range Digital Elevation Map (DEM)s (in the order of meters to few tens of meters in range). Because of the entailed limited situational awareness, especially regarding areas at a distance of the rover or occluded by rocks, this demands very conservative planning, as can be seen in the field report (Azkarate et al., 2016). We highlight distant areas, because this is where stereo images will struggle to deliver good disparities and the resolution of DEM products is significantly lower. Considering the high value of the rover system on Mars, it is well understood that if there are any potential risks at a mid to far distances that cannot be properly evaluated, the team will be hindered from commanding the rover further these risks and therefore command it to stop in front and wait for the next day, so that the conditions can be reevaluated from a closer distance. While this mode of operation is safe and might be quick in easy terrain, it heavily reduces the rover's range as the terrain becomes more rugged.

The necessary rover capabilities to implement this navigation mode are on board relative localization and a path follower or trajectory control module such as the one explained in subsection 3.1.

2.2 Path Following with Emergency Stops (Hazard Detection)

Building on top of the previous approach, we suggest to add a safety feature to allow for less conservative planning while minimally increasing the level of autonomy and having only little impact on the computational

cost.

The proposed hazard detector (Section 3.2) identifies possible obstacles, rocks or craters, and alerts the rover to stop the motion. This detector only searches for hazards in close vicinity of the rover front wheels ($< 0.5\,\mathrm{m}$) and acts as a "electronic bumper". The reason for this is twofold: it reduces the computational load for implementing the detector and is more flexible to allow more maneuvers, i.e., the rover will not stop if the hazard is 3 meters in front, and the detection is only triggered when the hazard falls inevitably onto the rover path. This feature, as simple as it may sound, is key to allow the operators to do their *Tactical planning* less conservatively, without increasing the risks for the rover.

2.3 Path Following with Hazard Avoidance

This navigation approach adds to the hazard detection feature, the capability of locally finding an alternative re-planned path, that allows to avoid the detected obstacle and continue with the original path as soon as the hazard is left behind. So, instead of just stopping in front of an obstacle, the detection of an obstacle triggers an additional path planner module (see subsection 3.3) that will execute an avoidance maneuver by providing to the trajectory control component a new list of navigation waypoints according to the re-planned path.

It is important to note that in this navigation mode, as long as there is no obstacles on the way, no additional computation is needed. That is, in an easy, non-hazardous terrain, this mode would be equivalent to the one following a pre-planned path with the hazard detector running on the background as an additional safety feature. On the other hand, when traversing a difficult terrain, this feature gives the operators team the possibility to command traverses of longer distance, where the limitation is no longer coming from risks that force a conservative planning but from the constrained resources of the system, i.e., the energy available for locomotion per Sol.

2.4 Fully Autonomous Navigation

While in the previous approach the rover can in theory have the capability to navigate autonomously and reach a certain commanded target on its own, the trajectory executed by the rover is potentially not the optimal one and would always be equal or longer than the one executed by a fully autonomous rover. The mapping processes running on board a fully autonomous rover allow it to asses the traversability of the terrain ahead from a better perspective and plan smooth and optimal paths. Additionally, the previous navigation mode still relies on a commanded path to follow that is prepared on ground. On the autonomous navigation approach though, the path to reach the coming scientific target is computed on board.

Fully autonomous navigation allows the operating team to focus on the *Strategic planning* and to select any target, be it in terms of coordinates or pixels of a reference picture. The team will trust the rover to plan a trajectory, identify obstacles on its own and adapt the trajectory as necessary. So far, this sounds like the ideal solution if it were not for the limited computational resources. This mode requires the on board execution of localization, mapping (or Simultaneous Localization and Mapping (SLAM)), path planning and trajectory control at least. For space, even heavily optimized algorithms can not deliver near-real-time computations while driving. Instead, as already mentioned in section 1, a rover would have to stop every 4-5 meters for tens of seconds in order to assess the situation and plan the trajectory for the next 4-5 meters. In relatively easy terrains this could be an unnecessary overhead, that can even rend the navigation less effective than a simple path following approach. However, as ExoMars and future exploration missions define more demanding requirements in terms of distance traverse and difficult areas to explore, the fully autonomous navigation capabilities will most certainly derive in hard system requirements.

3 Building blocks for efficient autonomous navigation

In this section we explain the three main components that we have developed for the implementation of the proposed efficient autonomous navigation mode with hazard avoidance capabilities. These are the trajectory control, hazard detection and local path planner components. As already mentioned our motivation is to propose a viable solution for autonomous navigation of Mars planetary rovers such as ExoMars, or future rover missions like the ones in Mars Sample Return. Taking the example of the ExoMars rover (see Figure 2) the sensor suite dedicated for navigation includes several optical cameras and/or stereobenches, namely Localization Cameras (LocCam), Navigation Cameras (NavCam) and Panoramic Cameras (PanCam). LocCam are the ones providing on board relative localization. This is typically done by means of a VO algorithm and fusing the inertial odometry estimation of the wheels and Inertial Measurement Unit (IMU). NavCam are used to get an overview of the terrain ahead and provide the operators with enough situational awareness of the rover surrounding, the NavCam images are commonly used for producing on ground stitched panoramas and DEMs. PanCam finally are actually considered scientific payload, but could also serve as a redundancy backup for the NavCam, given their similar characteristics and position.

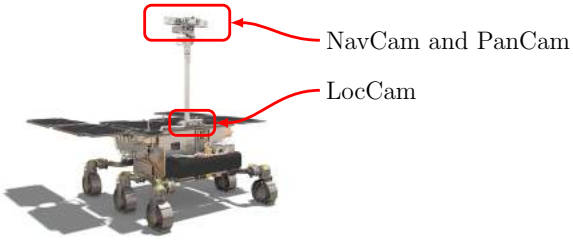


Figure 2: The location of the scientific and engineering stereo cameras on ExoMars (Credit ESA).

The control architecture approach proposed in this paper makes use of the LocCam and the products of the VO component running on board. As seen in Figure 3, the path planned by ground control operators is fed to the trajectory control component in order to follow it and to the local path planner to keep it memory. On the other hand, the LocCam provides stereo images to the VO component, which process them and provides continuous relative localization that is needed for the trajectory control. During the pose estimation process VO also computes the stereo disparity, which is fed to the hazard detection module. This module is constantly checking for hazards, and as soon as it detects one it sends the registered hazard to the local path planner. This is done in the form of a traversability grid containing 1s and 0s to indicate the existence of hazards in the corresponding area. This triggers a re-planning task that uses the latest estimated pose, the original planned path and the information of the registered hazard to compute a new path that avoids the named obstacle and rejoins the original path soon after it.

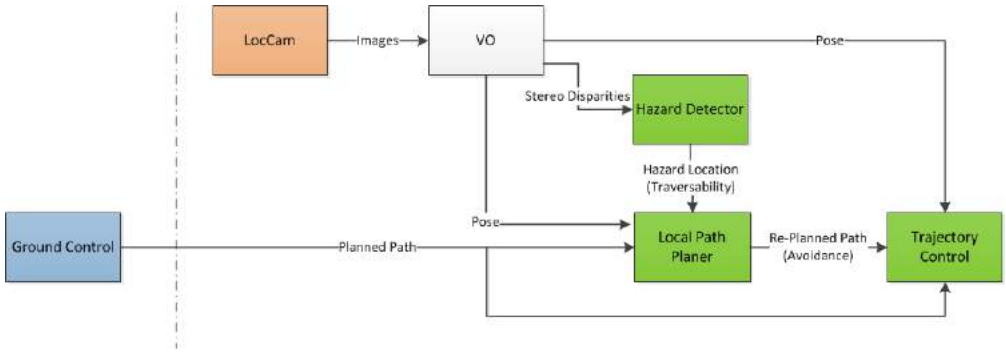


Figure 3: Conceptual schematic overview of the efficient autonomous navigation control architecture

It is worth mentioning that the proposed architecture runs with almost no computational overhead for most of the time compared to the current ExoMars navigation approach, with only the hazard detection check

running in parallel. The efficient algorithm explained below needs no major additional computations for detecting hazards, making use of data processed by the VO component. Only when a hazard is detected the local path planner is alerted to trigger the re-planning. As long as no hazards are detected the path planner is silently waiting. Similarly, as soon as the re-planning is done after a hazard detection, the path planner goes back to waiting mode.

The three subsections below explain in more detail how these components are implemented.

3.1 Trajectory control

In this section we explain the algorithm employed for controlling the trajectory executed by the rover. This will be used to follow a given path, whether this is the global path towards the next navigation target or a locally re-planned path when a hazard is being avoided. A path is basically a list of points in the operational Cartesian space (called waypoints) which the robot should visit in order. Thus, global and local paths are seamlessly treated given their equal structure and their only difference is the distance between consecutive waypoints (length of segments), global path waypoints being typically more separated in space. The task of trajectory control, also referred to as *path following*, consists of steering the vehicle along the path to the final waypoint or goal position. The path is provided by a higher-level component such as the on-board path planner (see subsection 3.3) or a human operator from ground, and is assumed safely traversable up to a certain deviation. This introduces the concepts of *nominal path*, comprised of the linear segments connecting all waypoints in the path, and *safety corridor*, an area bounded by two parallel lines, to the left and right of the nominal path at a certain symmetric distance. The *tracking error* is defined as the lateral deviation from the nominal path, measured from the rover position to the nearest segment line.

The trajectory control described in this paper is based on the popular *pure-pursuit* path follower algorithm (Coulter, 1992). This algorithm was used for trajectory control of cars by successful teams both in the DARPA Grand (Urmson et al., 2005) and Urban Challenges (Patz and Harper, 2008; Buehler et al., 2009), as well as for planetary rover trajectory control (Helmick et al., 2006). In the development of our algorithm, named *c-pursuit* (or *conservative-pursuit*), we modified the original approach to ensure that the path following was bounded within a predefined safety corridor, which corresponds to limiting the maximum allowed tracking error (ϵ). This is considered a critical safety feature for planetary exploration missions. Additionally, being a geometry-based algorithm, its parameters are intuitively configurable, as they depend on physical characteristics and capabilities of the rover. Finally, the algorithm does not require the step of generating a smooth interpolating trajectory, which is commonly done in path control algorithms. Instead it guides the rover along the path allowing deviations at turns to smoothly transition between consecutive segments.

3.1.1 *c-Pursuit*

This algorithm is developed with focus on rovers for planetary exploration. Rovers typically move at low speeds in the order of cm/s. Therefore, dynamics equations may be neglected and the problem is represented using kinematics equations. We consider a typical rover locomotion architecture, such as the one of ExoMars, with six driving wheels but with only four steering drives, as the lab rovers used in our experimental validation have fixed center wheels. Yet, this allows to perform motions of the type of Ackermann and spot turns, which provides enough dexterity to follow any kind of trajectory in the 2D plane. Ackermann maneuvers are in general more efficient than spot turns, as the latter requires the rover to stop and align the steering drives before turning while the former can be performed without stopping, adjusting the steering angles continuously as it moves forward.

The Ackermann steering of a six wheeled rover with four steering wheels can be thought of as a bicycle 2D kinematics model where the bike's rear wheel represents the center of the rover body and the bike's front wheel is in the center of the two front wheels of the rover. Given the rover is moving on a surface, its state is represented with the 2D position coordinates $x_r = [x, y]$ and its orientation (or *heading*) θ.

The original *pure-pursuit* algorithm solves the path following problem of an Ackermann-steered vehicle by providing the turning radius (or radius of curvature) that will drive the system along the path. Given a certain translation speed, the rotational speed is also fixed and the Ackermann maneuver to be executed is uniquely defined. The main steps of the algorithm are:

1. Determine the position of the vehicle x_r with respect to the nominal path.
2. Determine a look-ahead point x_{lh} ahead on the path.
3. Calculate the radius r_{turn} required to steer the vehicle from x_r to x_{lh}.

where the *look-ahead point* is a point in the nominal path that is found at a *look-ahead distance* (d_{lh}) from the rover position in the direction of the motion. The vehicle is instantaneously steered toward this virtual point. As the vehicle moves, the *look-ahead point* also shifts forward, hence the name pure-pursuit.

In *c-pursuit*, instead of having a constant parameter for the *look-ahead distance*, it is dynamically adapted depending on the tracking error ϵ of the rover at each control iteration.

$$d_{lh}{}' = d_{lh} - k_\epsilon \cdot \epsilon \tag{1}$$

where d_{lh} is the fixed parameter setting and $k_\epsilon > 0$ is the gain of the deviation penalization. Lowering the value of the *look-ahead distance* has the effect of bringing the rover back to the nominal path faster, with a sharper turning radius. In the extreme cases where the curvature exceeds the steering limits, a point turn is commanded to realign the rover with the heading towards the *look-ahead point*. Additionally, in *c-pursuit* the *look-ahead point* is not found in a circle at a *look-ahead distance* but instead is at a distance $d_{lh}{}'$ along the path from the projected position x_i. Figure 4 illustrates the modifications introduced by *c-pursuit* to the original *pure-pursuit*.

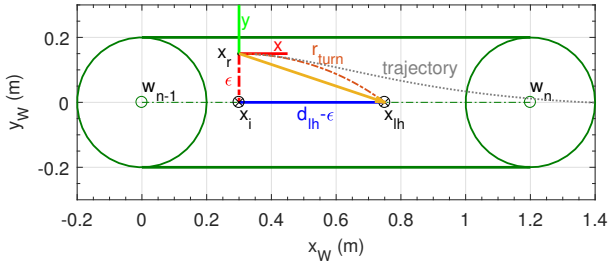
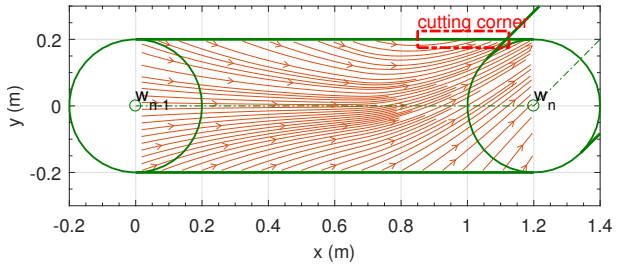


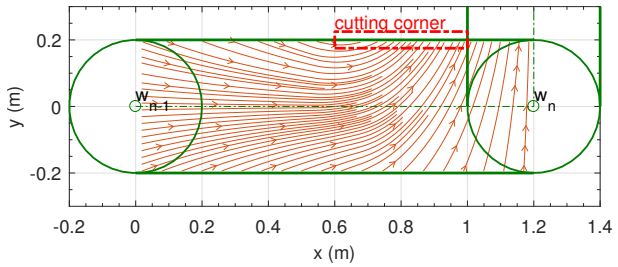
Figure 4: *Look-ahead point* x_{lh} calculation at position x_r with $d_{lh} = 0.6\,\text{m}$, $k_\epsilon = 1$ and $\epsilon = 0.15\,\text{m}$. w_n refers to the Cartesian coordinates of the path waypoints.

As the vehicle moves, the *look-ahead point* shifts along the path and this control scheme results in the trajectory shown in gray, which gradually converges back towards the path while progressing along it. Interestingly, the modifications added by *c-pursuit* have a relevant impact when two connecting segments form an increasing angle. Both algorithms allow a certain deviation from the nominal path, i.e., "cutting" corners, and guide the rover through smooth trajectories with Ackermann maneuvers (instead of sequences of straight lines and point turns). However, *pure-pursuit* can potentially guide the rover towards the outside of the safety corridor, specially when the change in heading between segments is greater than 45°. We define here γ as the angle formed between two segments, that represents this change in heading. As can be seen in Figure 5, *pure-pursuit* already cuts the corners to the point of exceeding the safety corridor for $\gamma = 45°$, while even at $\gamma = 90°$ *c-pursuit* is always directing the rover "inwards".

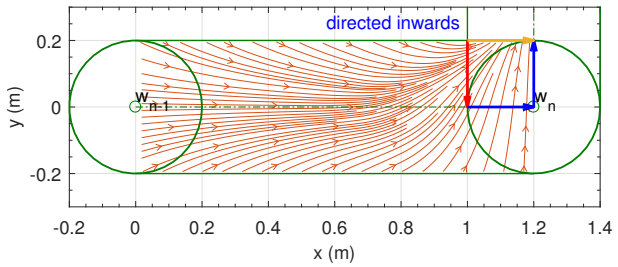
In (Snider, 2009) it is demonstrated that the *pure-pursuit* algorithm behaves like a purely proportional, or P, controller with the P gain being inversely proportional to the *look-ahead distance*. In *c-pursuit* the same



(a) original pure-pursuit algorithm, $\gamma = 45°$ between segments



(b) original pure-pursuit algorithm, $\gamma = 90°$ between segments



(c) proposed modification, $\gamma = 90°$ between segments

Figure 5: Vector fields of direction vector to *look-ahead point* x_{lh} from all possible x_r inside the safety corridor

applies, but in this case the P in an adaptive gain depending on k_ϵ. By appropriately selecting the values of d_{lh} and k_ϵ we can ensure convergence with nominal path and satisfy the corridor-bounded tracking of the path, which is the main motivation for the development of a trajectory control algorithm for planetary rovers. An easy guideline that guarantees this safety feature while leading to a smooth motion is to start by setting the *look-ahead distance* in the range of $d_{lh} \in \langle r_{\text{turn,min}}, 1.5 \cdot r_{\text{turn,min}} \rangle$ to allow sufficient changes in direction without reaching saturation of the commanded turn radius too quickly (see Figure 6). With $k_\epsilon \geq 1$, the safety corridor distance is set to $d_{\text{corridor}} = \frac{2}{3} d_{lh}$, which allows a path tracking error of up to $\pm\frac{1}{3} d_{lh}$ and drives the rover towards the nominal path (see 5c).

This controller has been implemented [1] and validated in experiments running onboard several ESA laboratory rover platforms, showing a robust performance in all cases. Main results of these experiments and further details of the described *c-pursuit* algorithm can be found in (Filip et al., 2017).

[1] Trajecory Control, `https://github.com/ESA-PRL/control-waypoint_navigation`

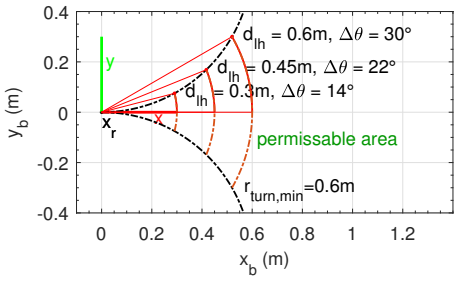


Figure 6: Minimum turn radius as the limitation on usable *look-ahead distance* value

3.2 Hazard Detection

As explained above, the hazard detection shall be computationally inexpensive and shall only rely on (engineering) sensors which are foreseen on the ExoMars rover.

We chose to develop the hazard detection based on the already existing stereo vision capabilities of the loccam located in the front of the rover. The necessary algorithms and calibrations for computing the disparities and distances are already present, since they are needed for visual odometry.

Because we want to develop the hazard detection as a safety feature which is fast to compute we do not want to compute more than what is necessary, e.g., we do not compute a 3D map representation of what lies in front of the rover, nor do we need to know what the next 10 or so meters look like. Instead, our aim is to create component which detects hazards that are right in front of the rover and allows the rover to quickly react and stop before driving into it.

These requirements allow us to have a small region of interest (RoI) with regard to the provided stereo images. In Figure 7, you can see a visualization of this RoI overlaid on top of the left camera's image.

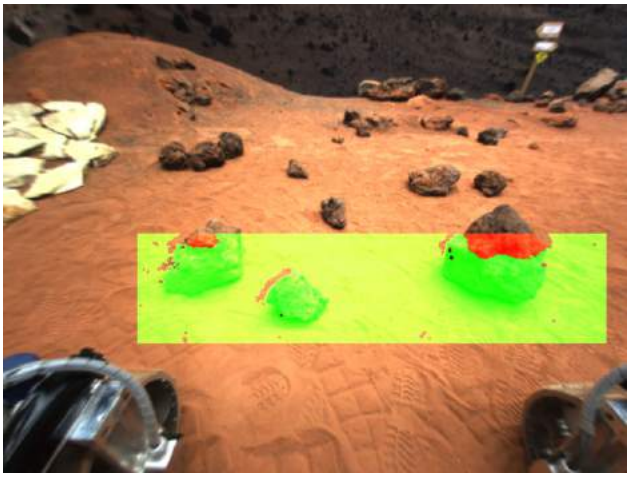


Figure 7: Hazard Detector Visualization: Green means safe, red pixels are considered a hazard, and pixels without an overlay are not regarded because they are outside of the RoI or because the disparity could not be computed.

3.2.1 General approach

Since the hazard detector is using a fixed camera (as opposed to the camera on the Pan-and-Tilt Unit (PTU)) and the capability to compute projective distances via this stereo camera is available, one can record which

distance values are to be expected at each pixel of the RoI when the rover is standing on and facing a flat surface. This yields a constant *calibration matrix*. This *calibration matrix* is actually computed over multiple averaged recorded samples to avoid spurious errors in the disparity calculation.

While driving, the rover is periodically computing the distances within the RoI and assumes that high deviations (be it positive or negative) from the calibration values are hazards.

Once it is decided that there is a hazard in front, the hazard detector has to compute in which position relative to the rover this hazard is located in order to output a traversability map so that the path planner can re-plan and initiate an avoidance maneuver.

3.2.2 Is there a hazard?

The hazard detector works based on few assumptions: The camera is mounted at a fixed position within the rover frame, it is facing forward, and it is tilted downwards at a fixed angle relative to the rover frame. As explained above, the nominal distance from the camera to a flat ground in front is known for each point $p_{\text{cal},i}$ within the RoI.

The stereo processing yields the distance from the camera to each point within the RoI and the hazard detector needs to decide whether the distance d_i for pixel i is within the tolerance.

Note that we cannot simply compare d_i to a constant tolerance value, e.g., 0.2 m, to recognize objects with a higher elevation than 0.2 m as hazards due to the pixel projection angle, because the distance is measured from the camera to the object (or ground). What this means is that comparing d_i to constant value for all pixels i would yield (projective) tolerances as depicted in Figure 8.

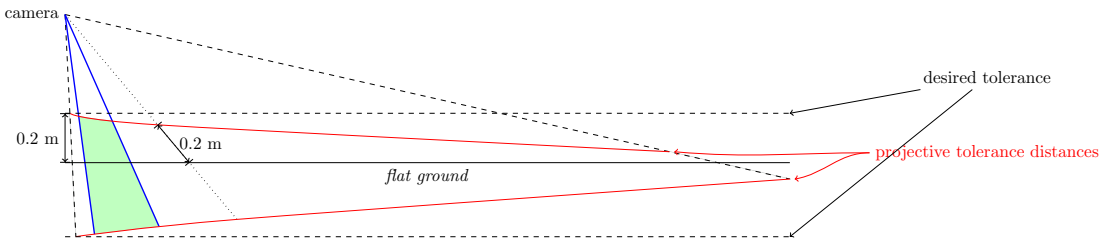


Figure 8: Conceptual side view of the camera with the field of view (dashed lines), tolerances (red), and region of interest field of view (blue).

From this information, two tolerance look-up tables can be computed – one for the hazards which are too close to the camera, e.g., rocks, and one for hazards which are too far away, such as cliffs or craters. Each of these tables contains for each pixel of the RoI, what the maximum or minimum tolerated distance is, such that the corresponding pixel would *not* be deemed hazardous.

Figures 9 and 10 depict how we can compute these tolerance values for one pixel. Let us first have a look at Figure 9 to see which information is necessary. For each pixel within the RoI, we know the calibrated distance $d_{\text{cal},i}$ from the camera to $p_{\text{cal},i}$. Let us construct a triangle defined by the camera's position, $p_{\text{cal},i}$, and the camera's position projected to the ground, (the red triangle in Figure 9).

Note that it is not important in which plane the triangle lies. This triangle is depicted in two dimensions in Figure 10. It is easy to see that we do not need any more information in order to calculate the tolerance values for that particular $p_{\text{cal},i}$.

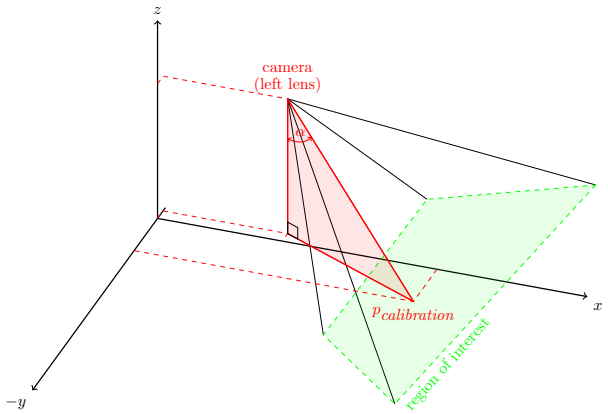


Figure 9: Position of the camera's left sensor relative to the rover's origin (at height 0), the region of interest (green), and an exemplary calibration point ($p_{\text{cal},i}$) used to determine the tolerated distances for the respective pixel.

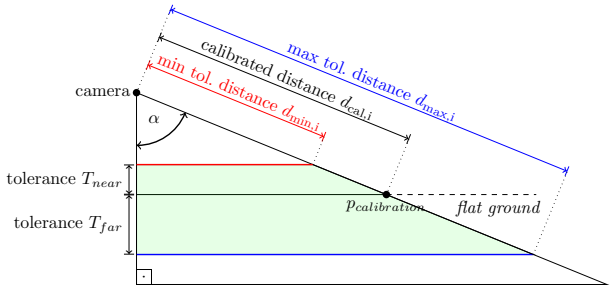


Figure 10: The tolerated distances from the camera are computed in such a way that hazards would be detected at certain heights.

Let h_{cam} be the height of the camera relative to the ground and $d_{\text{min},i}$ ($d_{\text{max},i}$) be the minimum (maximum) tolerated distances to be computed for $p_{\text{cal},i}$, while $d_{\text{cal},i}$ is the distance for pixel i in the calibration matrix and T_{near} (T_{far}) is the threshold distance from the groundlevel before a pixel is considered too close (far).

We can compute the cosine of α from the known camera height and $d_{\text{cal},i}$:

$$\cos(\alpha) = \frac{h_{\text{cam}}}{d_{cal,i}} = \frac{h_{\text{cam}} - T_{\text{near}}}{d_{\text{min},i}} = \frac{h_{\text{cam}} + T_{\text{far}}}{d_{\text{max},i}} \tag{2}$$

As can be seen in Equation 2, the minimum and maximum tolerated distances for i follow immediately:

$$d_{\text{min},i} = \frac{h_{\text{cam}} - T_{\text{near}}}{\cos(\alpha)} = \frac{d_{\text{cal},i} \cdot (h_{\text{cam}} - T_{\text{near}})}{h_{\text{cam}}} \tag{3}$$

$$d_{\text{max},i} = \frac{h_{\text{cam}} + T_{\text{far}}}{\cos(\alpha)} = \frac{d_{\text{cal},i} \cdot (h_{\text{cam}} + T_{\text{far}})}{h_{\text{cam}}} \tag{4}$$

After computing Equation 3 and Equation 4 for every pixel i in the RoI (which can be done as part of the calibration procedure), every distance computed by the stereo processing component can be compared to these two thresholds in order to decide whether they are hazardous or not.

Hazards will span over multiple pixels, which is why we can introduce a threshold for a minimum amount of pixels which have to be hazardous. Only if there is a large enough number of pixels in the RoI for which the hazard detector decides that they represent hazards, it will notify the rover so that it stops.

In case hazardous pixels are spurious due to, e.g., difficult lighting conditions, the hazard detector records multiple frames and rejects those pixels which do no appear as hazards on a minimum amount of samples, which is a simple and efficient mechanism to "filter outliers".

3.2.3 Where is the hazard?

At this point, the hazard detector algorithm has decided that a large enough number of pixels in its RoI represent hazards and it has discarded the possibility of those pixels being mere artifacts of the sensor or the stereo processing. The rover has safely stopped in front of the hazard and could wait for new instructions from ground.

Alternatively, we introduced the capability to autonomously avoid said hazards. But so far, we do not know where the hazard is in relation to the rover (other than "somewhere in view of the camera in front").

To find the corresponding locations in the real world, we compute the perspective transformation matrix from the stereo image to the ground plane.

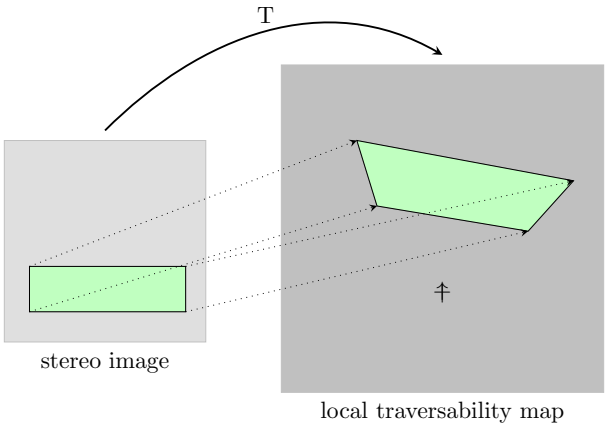


Figure 11: Transformation from the stereo image pixels to the local traversability map.

This way we can compute the two dimensional perspective transformation T from any pixel within the stereo image's RoI to a coordinate in the local traversability map (see Figure 11).

To project from point p_0 in the stereo image to point p_1 in the traversability map, we have to find T, such that:

$$\underbrace{\begin{bmatrix} x_1' \\ y_1' \\ w_1' \end{bmatrix}}_{p_0} = \underbrace{\begin{bmatrix} m_{00} & m_{01} & m_{02} \\ m_{10} & m_{11} & m_{12} \\ m_{20} & m_{21} & 1 \end{bmatrix}}_{T} \cdot \underbrace{\begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}}_{p_0} \tag{5}$$

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} \frac{x_1'}{w_1'} \\ \frac{y_1'}{w_1'} \end{bmatrix} \tag{6}$$

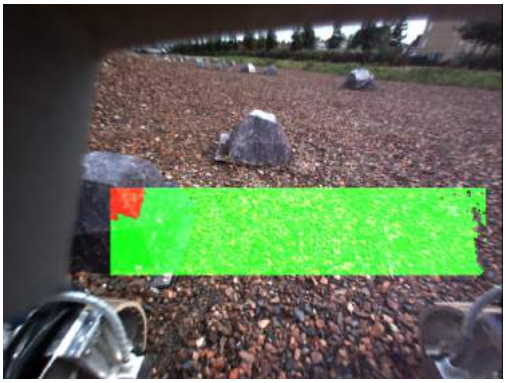
$$x_1 = \frac{m_{00}x_0 + m_{01}y_0 + m_{02}}{m_{20}x_0 + m_{21}y_0 + 1} \quad \wedge \quad y_1 = \frac{m_{10}x_0 + m_{11}y_0 + m_{12}}{m_{20}x_0 + m_{21}y_0 + 1} \tag{7}$$

As T has eight degrees of freedom ($m_{i,j}$), at least four pairs of corresponding points, introducing two constraints each, have to be found (i.e., no three points can be co-linear). More details for this standard problem in CV can be found in (Kindlmann, 2018).

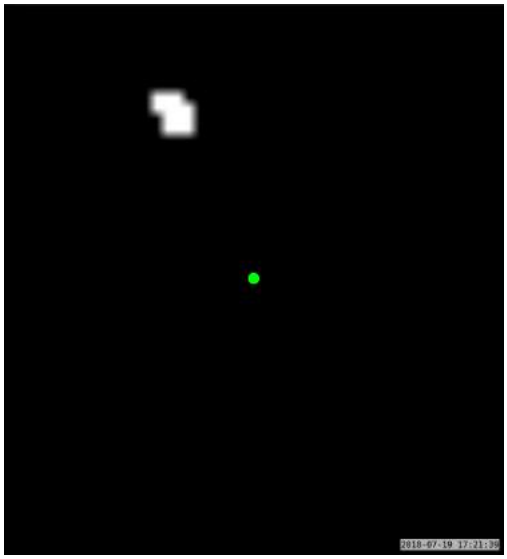
Because they are easy to locate, we measure the coordinates of the RoI's corners when projected onto the ground in the rover frame. To this end, we let the hazard detector's visualization run (see Figure 7) and place markers in the corners of the RoI. We can then find the x and y coordinates of those markers in relation to the rover frame.

The transformation matrix has to be computed only once (it can even be stored in a calibration file). Afterwards, for each hazardous pixel, we simply transform the pixel indices to the rover frame coordinates.

Figure 12a depicts the stereo image (with overlay) from which the local traversability map in Figure 12b is computed.



(a) Hazard Detector Visualization.



(b) Traversability Map. Hazards are marked as white pixels. The rover is in the center of the frame facing upwards (green dot).

Figure 12: An example for a detected hazard in the visualization in (a) and the corresponding, rover-centric, local traversability map in (b). The green dot denotes the rover position.

Note that this map is in the rover frame, while the path planner we use (see subsection 3.3) expects traversability maps in the global reference frame. Additionally, the planner does not have a model of the rover representing its dimensions. Instead, a rover is modeled as a single pose on the map. In order to actually avoid hazards, the local traversability map has to first be dilated and rotated into the global reference frame.

3.3 Local Path Planner (Hazard Avoidance)

By detecting a hazard that was not previously considered, the rover acquires more data relative to the environment in which it is moving. Indeed, the path it was following until that moment was originally created

without considering this new information. This may introduce a new problematic, since the immediate next waypoints could be located in unsafe places. Therefore, a local path planner is used here to dynamically update the path according to the new received information. It is assumed that only those waypoints from the original path that are closer to both the hazardous element and the rover itself need to be modified, since the rest of them are either already visited or unexplored. In other words, with this solution the rover only re-computes on board the immediate next waypoints, producing as result a local section of path that replaces them.

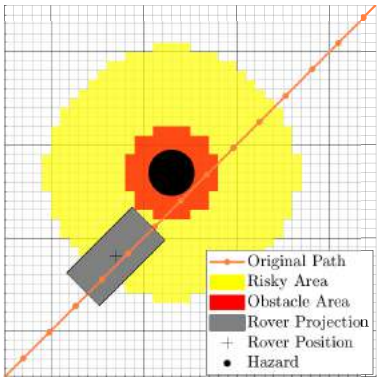
To better clarify how the local path planning component works, a graphical example is provided in Figure 13. It describes an hypothetical case where the rover is following a path and suddenly has the need to repair it due to the presence of a hazardous element detected on its way. The first step consists on taking the information provided by the local traversability map, such as the one shown before in Figure 12b, to update the grid. As seen in Figure 13a, the grid is formed by square nodes placed regularly over the map. Depending on the location of these nodes with respect to the detected hazard they can fall into three different kinds of area: *Obstacle Area*, *Risky Area*, and *Safe Area*. Unlike the last two ones, the *Obstacle Area* contains those nodes that are not traversable by the rover according to the local traversability map. With respect to the *Risky area*, it is in fact a traversable area, but as its name denotes, special care is given to this area since it contains nodes that are close, within a certain distance, to the *Obstacle Area*. Finally, any other *Local Node* located outside of the previous two mentioned areas falls into the *Safe Area*. This distinction is key to define the cost function that is later used by the local planner.

The method used to obtain the local waypoints is the Fast Marching Method (FMM), in particular its heuristic version (FM*). FMM was originally introduced by (Sethian, 1999) and used to numerically compute the expansion of a wave through a continuous 2D surface (extendable to 3D). As result, it provides a potential field from which the path connecting two points can be extracted by means of the gradient descent method. One of the main advantages of this algorithm is its speed, having a computational cost of $O(\zeta log\zeta)$, where ζ is the total number of nodes. Besides, its use in path planning applications (Liu and Bucknall, 2015; Garrido et al., 2016) is appreciated thanks to its capability to provide smooth, continuous and optimal paths. They are smooth in the sense that they do not present sharps edges or turns, mainly due to the fact that they are not limited to a discrete range of turning angles like in other path planning algorithms, e.g. D*. Besides, they can pass through node edges and are not limited to passing through node centers. Resulting paths are basically continuous single curves connecting two points in the 2D grid that integrate the least possible amount of cost, and thus they are the optimal ones. At the expense of obtaining quasi-optimal paths, an heuristic function can be used as shown in (Petres et al., 2005), making the computation even faster since it is prioritized to expand the wave towards the location of a certain point, e.g., the vehicle position in case the wave expands from the goal. Since for this particular case it is prioritized to lower the amount of computational resources needed, FM* is used instead of FMM.

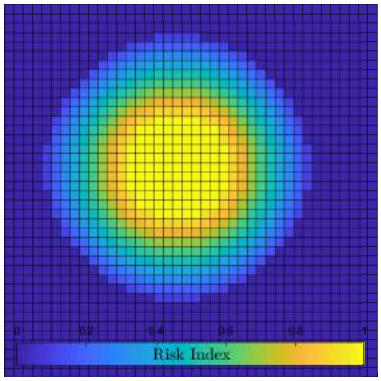
Cost used by FMM/FM* comes in the form of a 2D matrix field that indicates the difficulty of traversing each node, and determines the rate at which the wave propagates over the surface at each point. In other words, the higher the cost is the slower the wave propagates. This dependency between the wave expansion and the cost is determined by an equation called *eikonal*. Such equation, adapted to the nomenclature used in this case, is provided in Equation 8. For any node n_{ij}, where i and j are its horizontal and vertical coordinates in the 2D plane, t_{ij} is the arrival time of the wave, and, as mentioned, its computation is dependant on the value of cost c_{ij}. The potential field to be obtained as result of executing FM* is formed by the values of t of each node. It is later explained how this equation is used on the grid.

$$||\nabla t_{ij}|| = c_{ij} \tag{8}$$

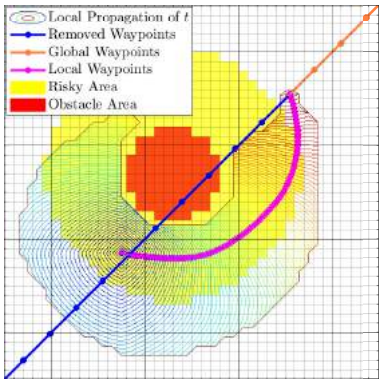
Prior to the execution of FM*, it is necessary to define the cost function c that sets the value of cost to each node. To comply with the requirements established before, this function must effectively make the resulting path to avoid the hazard by getting further from it. The way to do this is by increasing the value of cost in those nodes closer to the *Obstacle Area*. In Equation 9 it is shown the expression used. Since it always



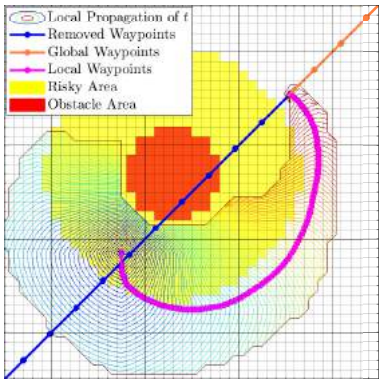
(a) An obstacle is found by the rover. The surrounding area is represented on a regular square grid and divided into *Obstacle Area*, *Risky Area* and *Safe Area*.



(b) A value of risk distance l from 0 to 1 is assigned to each node depending on the area in which they are contained.



(c) Resulting path using $r = 1$. This path goes through *Risky Area* since it is still traversable and gets further enough from the hazard to avoid colliding with it.



(d) Resulting path using $r = 2$. As can be observed, the path starts with a pronounced turn since it is prioritized to get further from the hazard.

Figure 13: Graphical example of the steps followed by the local path planner. In this case the rover finds a hazard on its way and plans an alternate path to avoid it.

returns a non-zero positive value, no local minimum points are created in the resulting potential field. Two parameters are here introduced: risk ratio r and risk distance l_{ij}. First of them determines the ratio between the highest possible cost (at *Obstacle Area*) and the least (at *Safe Area*). It is proportional to the gradient of the cost values, i.e., the rate at which the cost of the nodes decreases as they are further from obstacles. Coinciding with the work presented in (Petres et al., 2005) relative to the use of curve constraints on path planning, increasing this gradient makes the resulting local path get further from the hazard in exchange of introducing more pronounced turns. This statement can be also checked in Figures 13c and 13d, where resulting local paths using different values of r are provided. Risk distance l_{ij}, associated to a node n_{ij}, is a parameter containing a value between 0 and 1 that is proportional to the proximity to the nearest obstacle. The criteria to determine its value is as follows: first of all, those nodes located inside the *Obstacle Area* and the *Safe Area* have their associated value of risk distance already determined, being it 1 and 0 respectively. Then, for the case of the nodes inside *Risky area* their respective value of risk distance is between 0 and 1 and depends on the distance that separates each of them to their closest obstacle, as in Figure 13b. Such value can be obtained by using either an average filter, similarly to (Petres et al., 2005), or by computing a preliminary FMM wave starting from the *Obstacle Area* nodes, like in the Fast Marching Square algorithm (FM2) used in (Valero-Gomez et al., 2013).

$$c(n_{ij}) = c_{ij} = (1 + rl_{ij}) \tag{9}$$

After having defined how the cost is obtained to be used along with Equation 8, the next and final step is to make use of this equation to compute the local propagation wave. In order to make its implementation more clear, the discretized version of this equation is shown in Equation 10. Here, λ represents the resolution of the grid, and tx_{ij} and ty_{ij} are the lowest values of t of the horizontal and vertical neighbors of node n_{ij} respectively. The condition set ensures the algorithm is still upwind, i.e., higher values of t are obtained as the computation advances and the wave propagates. This is because the quadratic equation could, in very rare cases, not provide a valid value of t, arising the need of using a Dijkstra expression instead to still comply with the upwind condition.

$$t_{ij} = \begin{cases} \frac{tx_{ij}+ty_{ij}+\sqrt{2(\lambda c_{ij})^2-(tx_{ij}-ty_{ij})^2}}{2} & \text{if } |tx_{ij} - ty_{ij}| \leq \lambda c_{ij} \\ \min\{tx_{ij}, ty_{ij}\} + \lambda c_{ij} & \text{otherwise} \end{cases} \tag{10}$$

As stated before, the waypoints computed by the rover must rejoin with the previous path as seen in Figures 13c and 13d. To do this, the propagation wave must start from the rover position and reach a certain global waypoint from the previous path, Γ. This waypoint is the one that is placed in the *Safe Area* immediately after those within *Risky* or *Obstacle Areas*. It is worth mentioning that by convention of the operators another waypoint placed even further could be chosen instead. The further that Γ is located the smoother the resulting path becomes, although more computation is required since the wave would have to propagate further. Once this waypoint Γ is selected, since in this case FM* employs the heuristic function seen in Equation 11, the computation done in this step gives more priority to those nodes that are closer to Γ, i.e., those with lower value of h_{ij}. In this function, $\overline{p}_{ij}$ is the position of the local node n_{ij}. FMM and FM* algorithms are not detailed further here to avoid redundancy and are thoroughly described in previously mentioned references.

$$h_{ij} = t_{ij} + |\overline{p}_{ij} - \overline{\Gamma}| \tag{11}$$

4 Experiments

In this section we present the results of the test campaign performed during July 2018, in a Mars analogue terrain located near the European Space TEchnology and Research Centre (ESTEC) establishment of ESA. The exact coordinates of the testing area are $52°12'57.9"N, 4°25'36.1"E$. The tests were performed over several days, culminating in two long traverses on the 18^{th} of 19^{th} of July, demonstrating the viability of the efficient autonomous navigation mode proposed in this paper for planetary exploration missions.

Prior to these successful long traverses, other shorter "unit test" were executed to validate individual components and tune their parameters. The most relevant unit tests are also presented at the beginning of this section.

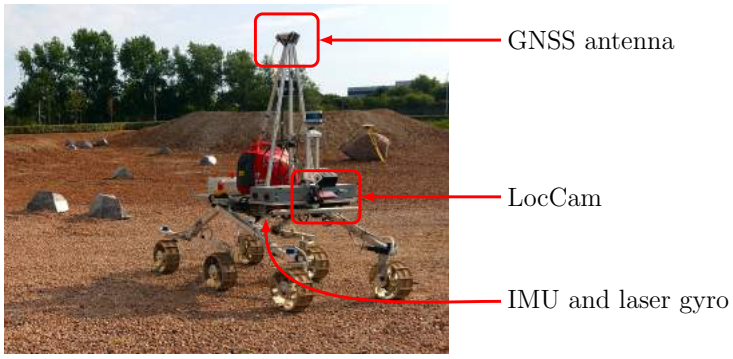
4.1 Field test objectives

The main objectives of the field test were (1) to understand whether a simple hazard detection setup and algorithm as proposed in subsection 3.2 may be sufficient to detect real hazards, (2) to verify whether it is possible to automatically avoid these hazards with minimal path corrections without staying too far from the commanded, original path, and (3) to demonstrate that in non-flat terrains with a medium distribution

of rocks, the combination of these two features (implementing the proposed autonomous control scheme), provides the ground operators team the possibility to perform longer traverses than what they nominally would, judging the risks from the received rover telemetry data.

4.2 Field test setup and system architecture

The rover used for this test is the Heavy Duty Planetary Rover (HDPR) from ESA's Planetary Robotics Lab (PRL) (PRL, 2018), on which we mounted a Bumblebee 2 (BB2-08S2C-25[2]) as the stereo camera in a location which is resembling the perspective of the LocCam on ExoMars (see Figure 14). The Bumblebee 2 has a wider field of view (FoV) than the original ExoMars LocCam but it is also mounted closer to the ground which eventually provides the mentioned similar viewpoint. For the rover localization, we opted to use a GNSS receiver with complementary Real Time Kinematic (RTK) corrections coming from a GNSS base station antenna, mounted at the field Control Station (Figure 16a). The reason for this selection was to focus the tests on the hazard detection and avoidance, in order not to affect the results by potential pose estimation errors. The state-of-the-art in VO can provide relative localization with an estimation error in the order of 1 to 2 % of the traversed distance. Considering our traverses have a length of 60 to 90 m, we believe our selection is justified. Using the RTK corrections we can actually reach absolute localization errors of less than 3 cm. The rover is also equipped with an IMU and an additional laser gyro sensor which are combined with the GNSS data to provide the full estimated rover pose, i.e. position and orientation.



(a) The stereo camera used for hazard detection is located in HDPR in a similar location as in the case of ExoMars's LocCam.

(b) The Bumblebee 2 stereo camera. The image is taken from (FLIR Bumblebee 2, 2018).

Figure 14: The rover and stereo camera used in the experiments.

The control station at the field that emulates the Ground Control is arranged inside a van that is also used for the equipment transportation (Figure 16b). Communication between the rover and the Ground is achieved by means of a long range WiFi antenna (Figure 16c). The control station runs the 3DROCS Software (Joudrier et al., 2013) for rover remote operations planning, monitoring, and control. This software is a branched version of the ExoMars rover visualization and planning software tool that will run at the Rover Operations Control Centre (ROCC) in Turin (Italy). Therefore, the installed control station mimics precisely the ground operators viewpoint in a real planetary mission. Finally, we used the SenseFly eBee professional 3D mapping drone (Figure 15) to acquire aerial images of the field test area and process them to generate DEMs such as the ones that an orbiting satellite, e.g. the Mars Reconnaissance Orbiter (MRO), would provide. This map can additionally be used by the operators for the *Strategic* and *Tactical planning*.

[2](FLIR Bumblebee 2, 2018)

Figure 15: The aerial drone used to map the test site is Sensefly's eBee Classic. The image is taken from (eBee Classic, 2018).

(a) RTK antenna placed on a location which is easy for geo-cross-referencing.

(b) A van is used to transport all equipment and to serve as a weather-proof base station.

(c) Long range wifi antenna for communication between the rover, the base station and thereby the RTK corrections.

Figure 16: The main components of the base station.

4.3 Validating the hazard detector

While the trajectory control and the path planner components had already been validated in several previous campaigns (Filip et al., 2017; Pérez-del Pulgar et al., 2017), the first tests of this campaign were dedicated to validating the novel approach for effective and computationally inexpensive hazard detection algorithm. HDPR's wheels measure 25 cm in diameter and the Rocker-Bogie passive suspension system is capable of overcoming obstacles of up to the wheel diameter. Adding some safety margin to this, we chose to set the tolerated height to 20 cm. Figure 17 shows a screen-shot of the hazard detector output, taken at the PRL. To the left of the detected rock we placed a ruler that we use to validate that any obstacle above the set threshold is identified as a hazard. In the picture this corresponds to 20 cm.

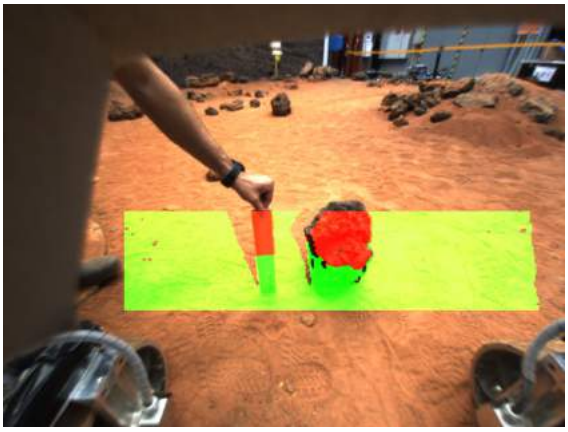


Figure 17: Different shaped rocks were used to verify that the hazard height was detected correctly.

Our next tests had the aim to check the impact in the detection algorithm produced by irregular surfaces, such as small hills or ripples, that can induce significant inclinations in the rover's *roll* and *pitch*. These tests were already performed in the field test area where this orography is naturally found. Three different cases were tested: (1) traversing a zone with multiple consecutive ripples, (2) going uphill up to the rim of a crater, and (3) descending from a hill with a small obstacle in front. The results of testing these three cases are shown in Figures 18, 19, and 20.

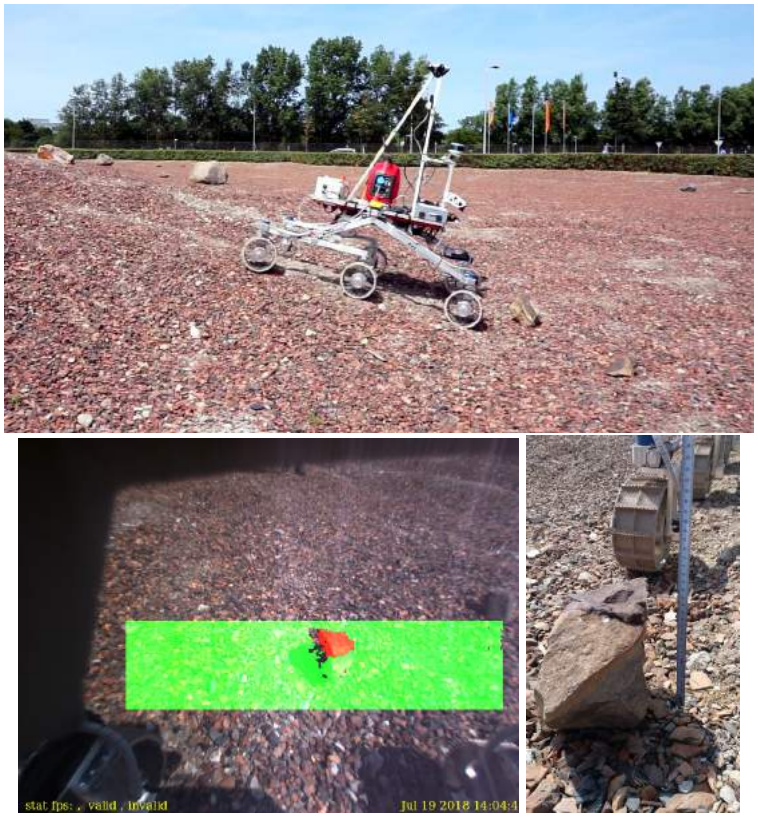


Figure 18: Rover descending from a hill towards a small rock.

Figure 19: Rover at the rim of a crater.

Figure 20: Rover traversing an area with ripples.

From these tests it can be derived that certain rover inclinations can potentially alter the outcome of the detection algorithm. Particularly, some scenarios that should in principle not represent a hazard can be falsely identified as hazardous because of the projection angle created by a significant rover inclination. In the first case, we see ripples being detected as hazards, and in case three we can see how a rock of 18 cm height is also detected as a hazard. Similarly in the second case, the rover identified the crater as a hazard, although in this case one could argue the crater represents a real hazard considering the steep downhill slope.

Although these cases can introduce false positive detections, they will not affect the safety of the rover and

just make the traversed path less efficient.

Interestingly, these initial test in the field revealed an additional unexpected issue. As can be seen in Figure 21 on bright days the sun can saturate some pixels, producing artifacts in the captured stereo pair which resulted in wrong disparity calculations. As the hazard detector relies on disparity calculations to compute the distance and compare it to a given threshold, this situation could produce false positives, making the rover avoid nonexistent obstacles. This is why we decided to cover the top and sides of the camera and to shade part of the sun rays inciding directly the camera. We also had to cover some of the metallic surfaces of the wheels, which are visible in the camera frame, with black duct tape, as their reflection would sometimes produce similar artifacts.

Figure 21: Sun artifacts. Difficult lighting conditions can render the computation of (correct) disparities impossible.

4.4 Efficient Autonomous Navigation Traverses

The main purpose of the increased autonomy and safety features introduced in this paper is to extend the reach of the rover into areas with poor visibility during the planning phase. That is to allow the operators to command the rover to traverse farther distances, even when the telemetry data does not provide sufficient detail (defined as the third objective in subsection 4.1). Two traverses were performed, in which an operator without those features would most likely choose not to advance very far, such as traversing through a valley (traverse 1) or advancing through an area with a lot of rocks (traverse 2). Most rocks used in the traverses are fake and manually set in the terrain in order to artificially create several hazardous situations along the traverses.

To demonstrate the added value of the proposed approach, the common procedure of a *tactical planning* phase is reproduced at the beginning of each traverse. To achieve this, a typical set of telemetry data is gathered from the rover sensors, particularly from cameras, as they provide the most valuable information for the operators evaluation of the environment and overall situational awareness. As can be seen in Figure 22, a stereobench with two FLIR Grasshopper (GS2-FW-14S5C[3]) cameras is mounted at this phase on top of the rover mast together with a PTU unit that allows for acquisition of images of the rover surroundings.

The set of images coming from the rover PanCam are received and processed at the Ground Control Station to produce stitched panoramic views and DEMs of limited range[4] (up to approximately 10 m). Figure 23 shows the view of the Ground Control Station at the beginning of the first traverse. We can reasonably assume that an operator cannot judge the risks of the terrain ahead after a certain distance (in the order of 10 to 20 m) due to the occluded view behind the two craters and probably would command the rover to navigate up to the entry point of the valley between these. However, using a global orbiter map, such as the one in Figure 24 with a resolution of 25 to 50 cm/pixel, operators can generate a roughly safe nominal path that takes the rover through this valley and reaches a further point in the map. Such map gives enough detail to avoid big obstacles or big risky areas (i.e. hills, craters) but does not reach the level of detail needed to avoid smaller obstacles like medium size rocks, such as the ones manually introduced.

[3](FLIR Grasshopper 2, 2018)

[4]The limitation comes from the fact that DEM quality decreases significantly with increasing distance.

Figure 22: HDPR with the mounted PanCam and PTU.

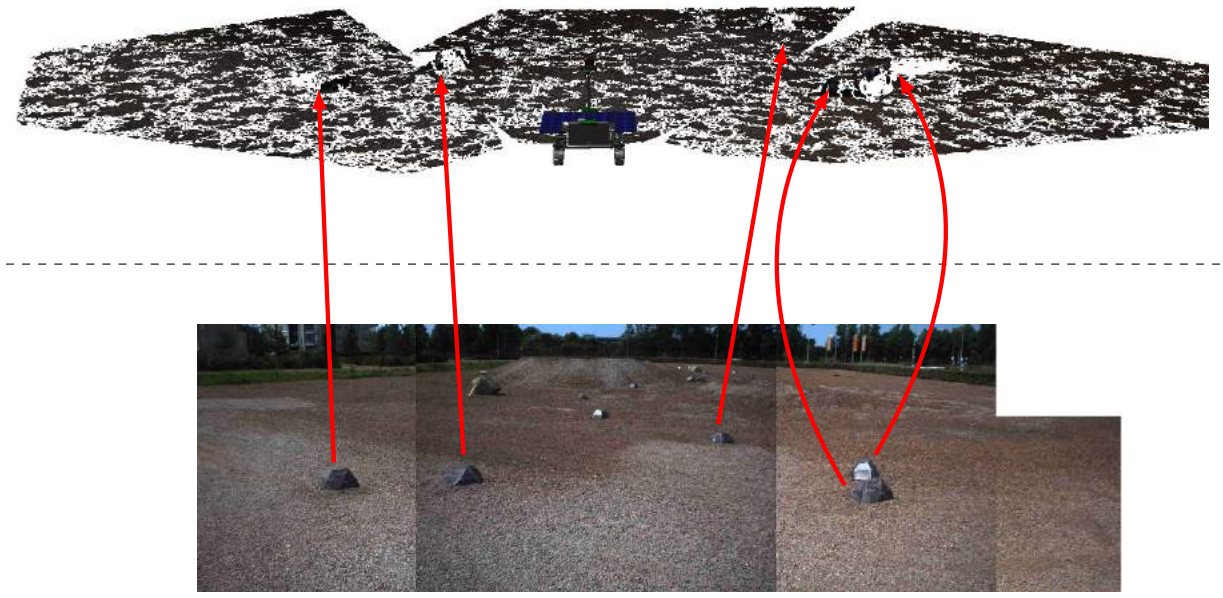


Figure 23: Assessment at the beginning of the first traverse. In the top you can see a DEM reconstructed from the stereo images composing the panorama in the bottom. Red arrows indicate corresponding hazard locations between the two. (The 3D model in front of the DEM is the 3D representation of another of our rovers.)

Figure 24: Orbital map of the test area. The image is a down-sampled version of the first test setup to approximate the resolution of HiRISE.

The rover was commanded to follow the planned path. Along the first traverse the rover was capable of successfully detecting all obstacles ahead, and performing avoidance maneuvers that would guide it forward rejoining the planned path once the hazard was passed. A false positive was noted, that did not have major impact in the overall traverse. Figure 25 provides a panoramic view of the performed first traverse through the valley between the craters. The resulting trajectory of the performed traverse can be seen in the high resolution aerial view in Figure 26.

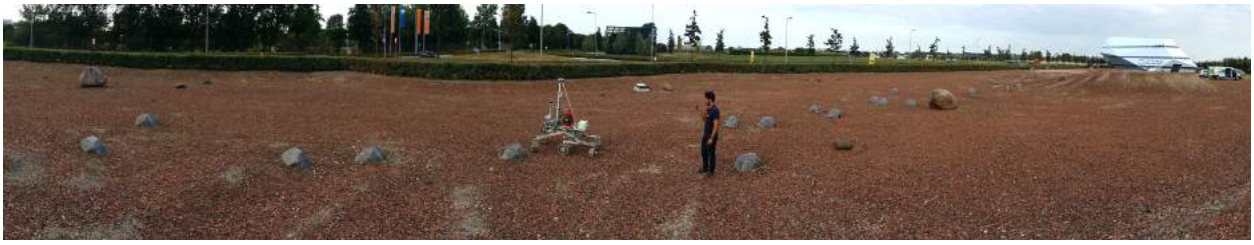


Figure 25: This panorama gives a good impression of the first traverse.

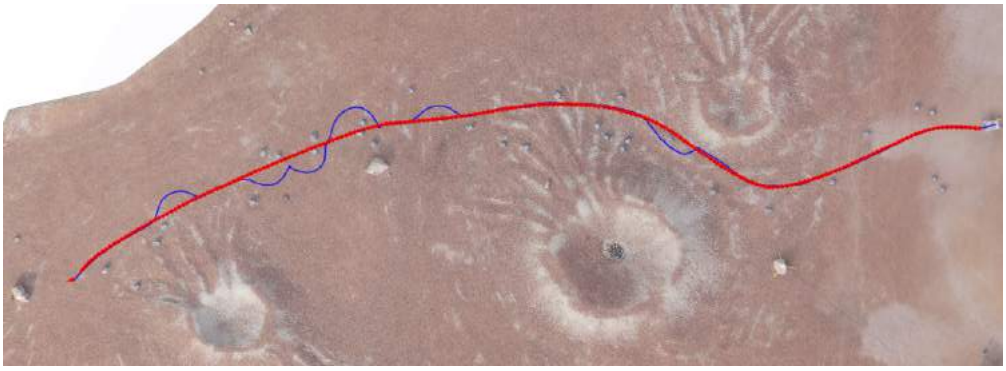


Figure 26: The first of our test setups. The red arrows represent the pre-planned waypoints while the thin blue line shows the actual traverse.

In a given situation though, the rover slightly "climbed" over the side of one of the rocks that had not been registered when passing next to it. This is not considered a failure of the hazard detector as the rock had not been seen within the hazard detector region. Additionally, in case of the rock being a real one, the rover would have had no risk to overcome it considering the height on that side of the rock. However, it was decided that for the second traverse the configuration of the local path planner would be modified to increase the penalty to the areas surrounding an obstacle (risk ratio) and therefore slightly increase the safety margin distance kept from a hazard.

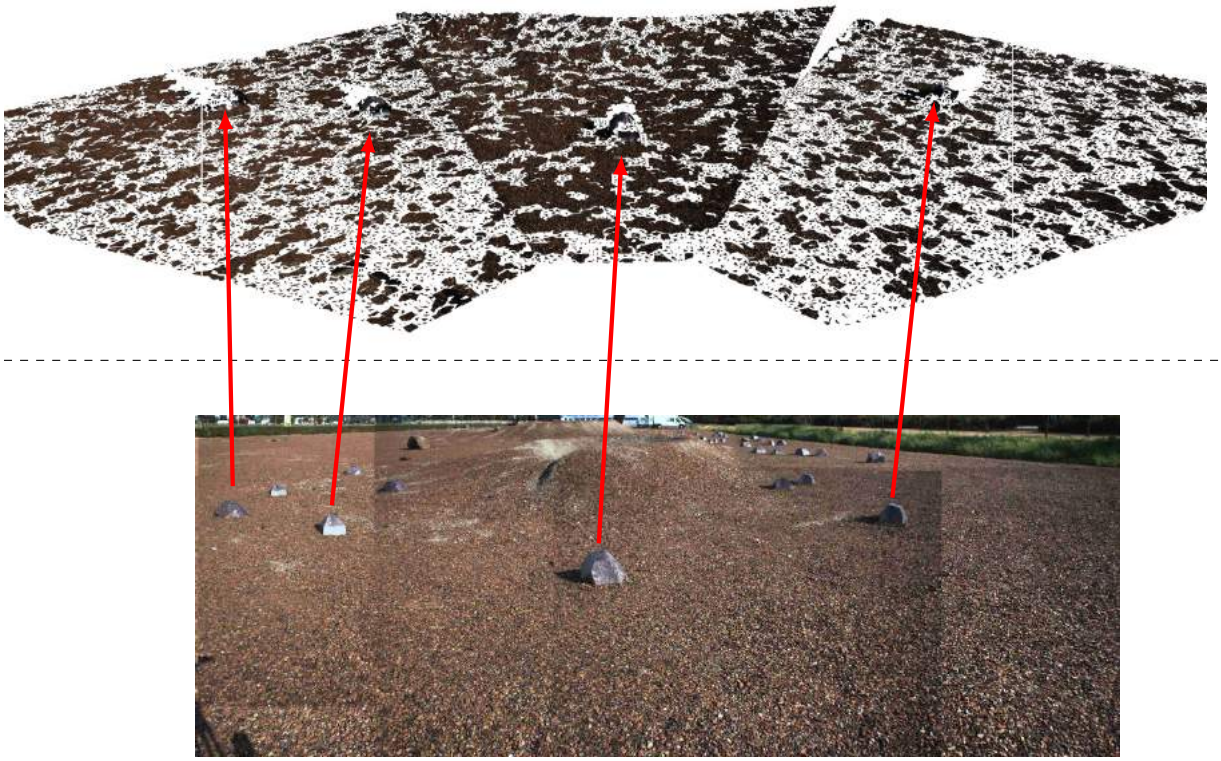


Figure 27: Assessment at the beginning of the second traverse. In the top you can see a DEM reconstructed from the stereo images composing the panorama in the bottom. Red arrows indicate corresponding hazard locations between the two.

The second traverse was performed following the same procedure. The images from the PanCam were gathered first for the *tactical planning* (see Figure 27).

Again, the telemetry data showed the difficulty of the terrain to traverse, making it impossible to generate a path that would be completely safe further than few tens of meters ahead. Using the global low resolution orbital map, a rough nominal path is generated and sent to rover to initiate the traverse (see Figure 28).

Figure 28: HDPR avoiding obstacles during the second traverse.

The rover successfully reached the final target, executing numerous avoidance maneuvers without registering any incidents, rendering this a success. Figure 29 shows the trajectory that resulted from the execution of the second traverse.

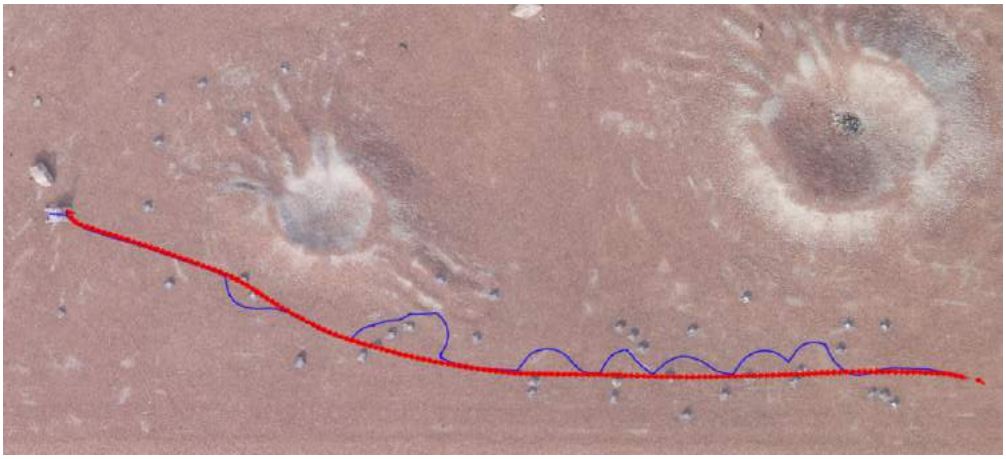


Figure 29: The second test setup. The red arrows represent the pre-planned waypoints while the thin blue line shows the actual traverse.

These results confirmed the achievement of the field test objectives. The three developed modules described in section 3 were experimentally validated and the proposed autonomous navigation methodology proved.

As explained in section 2, the approach for efficient autonomous navigation tested in this campaign provides

a solution for a computationally cheap navigation scheme. As expected, the trajectories that result when running with this control architecture are longer, in terms of total traversed distance, than the nominal paths originally planned, given the re-planning of the trajectory for hazard avoidance happens at the local level. This leads to globally suboptimal trajectories. In the case of the first traverse, the nominal path had a length of 86 m while the performed trajectory had a length of 97.7 m, which implies an increase of 13.6 %. In the second traverse, the nominal path had a length of 62.6 m and the performed trajectory had a length of 79.2 m, an increase of 26.5 %. While the total traversed distance could potentially be lowered by running a fully autonomous navigation scheme, we believe the computational simplicity of our control architecture outweighs the extra distance. These scenarios had a low to medium amount of rocks distributed along the path. In scenarios with higher amount of risks or more complex terrains, where re-planning would be more frequently triggered, an approach with full mapping and navigating capabilities might be more convenient.

4.5 Potential risks, limitations, and possible improvements

The initial tests performed in the field to validate the hazard detector already showed some cases where the detector could potentially signal false positive hazards. While this does not represent a safety issue it can render the control inefficient if many of these were to happen along the traverse. A solution to reduce this, could be to consider the actual rover orientation in the calculations by including the IMU data in the detection algorithm. Additionally, one could consider that, in cases of high inclination, the hazard detection could trigger a full mapping procedure (3D reconstruction) and an overall traversability evaluation before running the re-planning task. Obviously this would require more time and computational complexity than the original approach. Actually, this can also be considered as a hybrid approach where the efficient and fully autonomous navigation modes are combined being the fully autonomous navigation mode started when a hazard has been detected while running in efficient mode.

Throughout the validation tests we noted the need to fulfill certain geometric constraints. The FoV of the LocCam and its perspective view (mounting height and orientation) need to allow for the hazard detector to cover a wide enough area in front of the rover. The width of the detection region needs to be equal to or bigger than the length (and width) of the rover. This allows the rover to register hazards at its sides when passing next to them and ensures the rover will not enter these hazardous zones (colliding with rocks or driving into a hole) when performing an avoidance maneuver triggered afterwards by another hazard in front.

5 Conclusion

We introduced the different navigation modes to operate a planetary rover and proposed a set of components implementing autonomous features that allow for an efficient autonomous navigation approach. These features are the trajectory control, hazard detection, and a local path planner for hazard avoidance. The experiments conducted during a field test campaign validated the proposed methodology, which enabled a planetary rover to successfully navigate two long traverses while avoiding hazards. The navigation method relies on a roughly safe path that is planned on ground based on rover telemetry and orbital data, and does not require continuous 3D reconstruction of the traversed terrain. When compared to the navigation and operation mode of the ExoMars rover mission, this approach enables the operators to plan less conservative tasks and complete longer distance traverses per sol, potentially increasing the scientific return of the mission.

The amount and frequency of hazards determines the difficulty of a terrain. The performed traverses had a low to medium distribution of rocks along the path. In such terrains, and given the computational and system limitations of typical planetary rovers, we stress the advantage of the proposed mode of operation over a fully autonomous rover. A fully autonomous planetary rover would typically need to stop every few meters in order to map the area in front and assess traversability, while rovers using our approach only need

to stop when they encounter a hazard. Depending on the difficulty of the terrain, the proposed solution can thus be more efficient.

References

Azkarate, M., Kapellos, K., Hewitt, R. A., Boukas, E., Joudrier, L., Poulakis, P., Bussi, D., Ferentino, E., and Visentin, G. (2016). Remote Rover Operations: Testing the ExoMars Egress Case. In 13th International Symposium on Artificial Intelligence, Robotics, and Automation in Space (iSAIRAS 2016), pages 1–8.

Bousquet, P. (2011). The CNES contribution to in-situ exploration: robotic implications. In 11th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA).

Buehler, M., Iagnemma, K., and Singh, S. (2009). The DARPA urban challenge: autonomous vehicles in city traffic, volume 56. Springer.

Cheng, Y., Maimone, M., and Matthies, L. (2005). Visual odometry on the mars exploration rovers. In IEEE International Conference on Systems, Man and Cybernetics.

Correal, R. and Pajares, G. (2011). Onboard autonomous navigation architecture for a planetary surface exploration rover and functional validation using open source tools. In 11th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA).

Coulter, R. C. (1992). Implementation of the Pure Pursuit Path Tracking Algorithm. Technical report, Carnegie Mellon University, Robotics Institute.

eBee Classic (2018). Retrieved August 27, 2018, from `https://www.sensefly.com/drone/ebee-mapping-drone`.

Filip, J., Azkarate, M., et al. (2017). Trajectory control for autonomous planetary rovers. In 14th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA).

FLIR Bumblebee 2 (2018). Retrieved August 16, 2018, from `https://www.ptgrey.com/bumblebee2-stereo-vision-08-mp-color-firewire-1394a-25mm-sony-icx204-camera`.

FLIR Grasshopper 2 (2018). Retrieved September 12, 2018, from `https://www.ptgrey.com/grasshopper2-14-mp-color-firewire-1394b-sony-icx285-camera`.

Garrido, S., Álvarez, D., and Moreno, L. (2016). Path planning for mars rovers using the fast marching method. In Robot 2015: Second Iberian Robotics Conference, pages 93–105. Springer.

Helmick, D. M., Roumeliotis, S. I., Cheng, Y., et al. (2006). Slip-compensated path following for planetary exploration rovers. Advanced Robotics, 20(11):1257–1280.

Jet Propulsion Lab (2018). High Resolution Imaging Science Experiment. Available at: `https://hirise.lpl.arizona.edu/index.php`. Last accessed on August 2018.

Joudrier, L., Kapellos, K., and Wormnes, K. (2013). 3D Based Rover Operations Control System. In 12th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA).

Kindlmann, G. (2018). Image mosaicing – background and principles. Last accessed on August 22, 2018, `http://www.cs.utah.edu/~gk/class/6964/1/tech.html`.

Kostavelis, I., Boukas, E., Nalpantidis, L., Gasteratos, A., and Aviles Rodrigalvarez, M. (2011). SPARTAN system: Towards a low-cost and high-performance vision architecture for space exploratory rovers. In IEEE International Conference on Computer Vision Workshops (ICCV Workshops).

Lentaris, G. et al. (2018). High-performance embedded computing in space: Evaluation of platforms for vision-based navigation. Journal of Aerospace Information Systems.

Liu, Y. and Bucknall, R. (2015). Path planning algorithm for unmanned surface vehicle formations in a practical maritime environment. Ocean Engineering, 97:126–144.

Maimone, M. (2017). The evolution of autonomous capabilities on NASAs mars rovers. In Southern California Robotics Symposium. https://www.youtube.com/watch?v=u4-4x8GhE6Y.

Maimone, M., Cheng, Y., and Matthies, L. (2007). Two years of Visual Odometry on the Mars Exploration Rovers. Journal of Field Robotics.

Matthies, L., Maimone, M., Johnson, A., Cheng, Y., Willson, R., Villalpando, C., Goldberg, S., and Huertas, A. (2007). Computer vision on mars. International Journal of Computer Vision.

McEwen, A. S., Eliason, E. M., Bergstrom, J. W., Bridges, N. T., Hansen, C. J., Delamere, W. A., Grant, J. A., Gulick, V. C., Herkenhoff, K. E., Keszthelyi, L., et al. (2007). Mars reconnaissance orbiter's high resolution imaging science experiment (hirise). Journal of Geophysical Research: Planets, 112(E5).

Merlo, A., Larranaga, J., and Falkner, P. (2013). Sample fetching rover (sfr) for msr. In 12th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA).

Moreno, S. (2013). CNES robotics activities: towards long distance OB decision-making navigation. In 12th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA).

Patz, B.J̃., P. Y. P. R. S. G. and Harper, D. (2008). A practical approach to robotic design for the DARPA urban challenge. Journal of Field Robotics.

Pérez-del Pulgar, C. J., Sánchez, J., Sánchez, A., Azkarate, M., and Visentin, G. (2017). Path planning for reconfigurable rovers in planetary exploration. In Advanced Intelligent Mechatronics (AIM), 2017 IEEE International Conference on, pages 1453–1458. IEEE.

Petres, C., Pailhas, Y., Petillot, Y., and Lane, D. (2005). Underwater path planing using fast marching algorithms. In Oceans 2005-Europe, volume 2, pages 814–819. IEEE.

PRL (2018). Planetary Robotics Lab. Available at: https://www.esa.int/Our_Activities/Space_Engineering_Technology/Automation_and_Robotics/About_the_Planetary_Robotics_Lab.

Sethian, J. A. (1999). Fast marching methods. SIAM review, 41(2):199–235.

Shaw, A., Woods, M., Churchill, W., and Newman, P. (2013). Robust visual odometry for space exploration. In 12th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA).

Snider, J. M. (2009). Automatic Steering Methods for Autonomous Automobile Path Tracking. Technical report, Carnegie Mellon University, Robotics Institute.

Urmson, C., Anhalt, J., Bartz, D., et al. (2005). A robust approach to high-speed navigation for unrehearsed desert terrain. Journal of Field Robotics.

Vago, J. L. et al. (2017). Habitability on early mars and the search for biosignatures with the exomars rover. Astrobiology.

Valero-Gomez, A., Gomez, J. V., Garrido, S., and Moreno, L. (2013). The path to efficiency: Fast marching method for safer, more efficient mobile robot trajectories. IEEE Robotics & Automation Magazine, 20(4):111–120.

Wilson, M. (2013). Mars 2020 Mission Concept. Available at: https://www.nasa.gov/sites/default/files/files/3_Mars_2020_Mission_Concept.pdf. Last accessed on August 2018.

Woods, M., Shaw, A., Tidey, E., Pham, B. V., Lacroix, S., Mukherji, R., Maddison, B., Cross, G., Kisdi, A., Tubby, W., Visentin, G., and Chong, G. (2014). Seeker—autonomous long-range rover navigation for remote exploration. Journal of Field Robotics.