VISION-BASED OBSTACLE DETECTION FOR PLANETARY ROVERS

Olivier Clerc[1,2], Levin Gerdes[2,3], and Martin Azkarate[2]

[1]*Robotic Systems Lab, ETH Zürich, Zürich, Switzerland*
[2]*ESA-ESTEC, European Space Agency, Keplerlaan 1, 2201 AZ Noordwijk, The Netherlands, Email: firstname.lastname@esa.int*
[3]*Department of Systems Engineering and Automation, University of Málaga, Málaga, Spain*

ABSTRACT

Communication and computing performance constraints of planetary rovers limit their traverse distances. Moreover, motion planning for these systems is a conservative matter. Indeed, such rovers are expensive and out of reach for physical help, and therefore, any collision during a mission critically endangers its success. Nevertheless, considering future missions such as Mars Sample Return - Sample Fetching Rover (SFR) that require daily traverses extending to hundreds of meters per Sol, there is a necessity for a high degree of autonomy. While autonomous navigation concepts have greatly improved for applications on Earth, they remain unsuited for planetary rovers. Considering our previous works on navigation architectures [3, 5], this paper proposes a real-time Obstacle Detector method to improve navigation efficiency by creating a medium to close range local obstacle map through the use of images from a stereo camera.

1. INTRODUCTION

Motion planning for planetary rovers is a conservative matter. Indeed, such rovers are expensive and are out of reach for physical help, and therefore, any collision during a mission critically endangers its success. Therefore, there is a need to plan the rover's path to reduce risks to the minimum. Moreover, for missions to Mars, direct driving is unfeasible. Not only does the round trip communication time between Earth and Mars takes from 5 to 20 minutes, but contact with ground can be reached only twice per Sol. Even with the help of orbital data and downloaded images from the rover, planning such long traverses with no obstacles on its route proves to be a practically impossible task. Considering the high precision of localization used in future missions such as the ExoMars rover, Townson et al. (2018) [11], it still suffers from inaccuracies from the absence of a Global Navigation Satellite System (GNSS). Moreover, the orbiter resolution is not high enough to detect all obstacles on the possible rover path. Considering future missions such as Mars Sample Return - SFR that require daily traverses extending to hundreds of meters, there is a necessity for a high degree of autonomy.

While autonomous navigation concepts have greatly improved for applications on Earth either using technologies like LiDARs and Machine Learning based algorithms, they remain unsuited for planetary rovers. These solutions come with significant drawbacks for efficient planetary exploration missions. One notable drawback is the high computational complexity of a robust mapping, localization, and path planning system, which is not compliant with the current capabilities of planetary rovers or would at least limit the effective mission lifetime due to additional stops for computations. Indeed, the high radiation levels on Mars are detrimental to sensitive modern processors; therefore, planetary rovers need specific types of fault-tolerant microprocessor architectures such as LEON [2]. For example, one of the latest rovers to have reached Mars, Perseverance, uses the same CPU architecture that powered the 1998 iMac G3 [1]. Therefore to safely navigate a planetary environment, a rover would need to drive slowly and regularly stop to solve the localization and motion planning problems for the next few meters ahead. As explained in our paper on rover Guidance, Navigation, and Control (GNC) architectures [3], a way to solve this issue would be to separate the autonomous navigation problem into two distinct situations using an initial low-resolution map of the rover's landing site created from previously acquired orbital data in the shape of satellite images. This map is used to give an initial path where the rover will avoid risky and non-navigable areas such as craters and large boulders. However, the map's resolution is too low to include small obstacles. Nevertheless, it can help determine specific characteristics of the traversability difficulty of the area where the rover should perform the traverses. These characteristics, in turn, can be used to classify sections of the traverses as either complex or benign. In 2019 [3], we proposed a two-level navigation system that relies on this difficulty distinction. A low-level *Efficient Navigation* concept designed for easy to moderate terrain difficulty and a high-level *Full Navigation* system for complex terrains as shown in Figure 1.

The *Efficient Navigation* system, therefore, aims to allow for faster and longer traverses in pre-assessed benign terrain and is the one for which this project proposes the developed obstacle detection algorithm. Ground control uploads an initial *Global Path* which can be, for example, two waypoints from the rover's current pose to the desired position at the end of the benign terrain or at the end of that Sol. With the help of the previously stated map, the produced path avoids the major obstacles on its

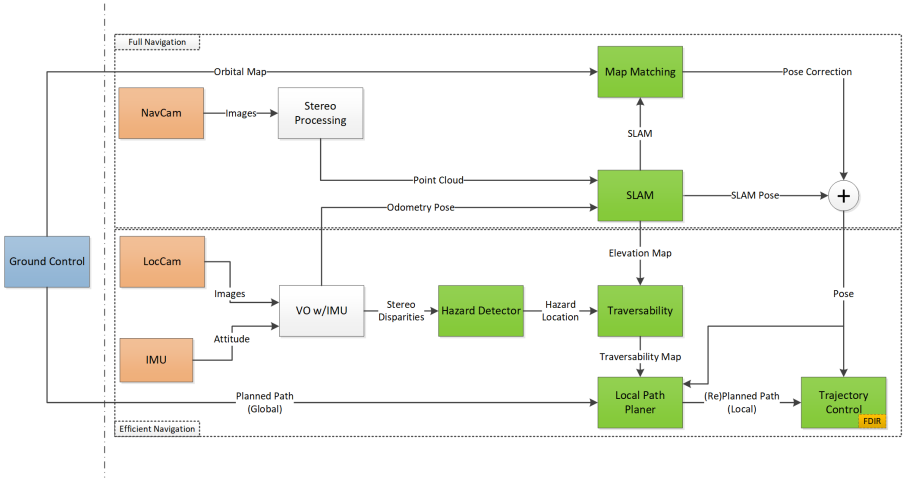


Figure 1. Schematic concept of the two-level GNC architecture for Autonomous Navigation of Planetary Rovers [3].

path. Using a stereo camera named LocCam and the Inertial Measurement Unit (IMU), the rover continuously performs localization to update its relative pose in its reference frame in order to drive along the path autonomously. However, the rover is still at risk of crashing into an obstacle missed in the low-resolution map. This issue is where the *Hazard Detector* comes into effect.

The hazard detector used in Azkarate et al., 2019 [3] that is in the center of the lower half of Figure 1 is presented in Gerdes et al., 2020 [5]. This proposed hazard detector relies on stereo images obtained from the LocCam. Using these images, the rover computes stereo disparities on a small region of interest to maximize efficiency and compares them to a pre-calibrated set of stereo disparity values for each pixel within the region of interest. These values are calibrated by measuring each pixel's stereo disparity on a flat floor. This method is highly efficient, but its major drawback lies in the flat ground assumption, which leads to false positives and negatives on uneven ground since the hazard detector considers neither the orientation of the rover nor the slope of the terrain. Such a system is only used for small ranges up to $50\,\mathrm{cm}$ [5] and can be interpreted, and was intended, more as an emergency safety feature. Such a small range is efficient to avoid crashing into an obstacle but can lead to less efficient paths than detection approaches with a greater range.

Therefore, this paper aims to improve the existing hazard detector in the sense of being still able to detect dangerous obstacles while improving navigation efficiency. As stated above, autonomous navigation concepts have improved and diversified over recent years, and there is a need to define the technology and type of algorithms used for this project.

Two types of sensors, laser-based, and cameras, are commonly used for obstacle detection. Even though laser-based technologies such as Time-of-Flight cameras and LiDARs offer a direct value of ranges that the rover's computer will otherwise need to compute on cameras with more complex algorithms such as stereo matching, they also have not yet been widely used in planetary rovers and have a low Technology Readiness Level (TRL). A lot of hardware development still has to be done, and thus they will not be used in missions such as ExoMars and SFR. Moreover, LiDARs consume much more power than cameras, and it is essential to note that power consumption is a deciding factor in planetary rovers. Camera sensors are also commonly used for autonomous navigation, and fields such as Visual Simultaneous localization and mapping (Visual SLAM) have matured over the years. These factors contribute to the choice of cameras, and more precisely, to stereo cameras that allow for easier 3D information extraction. The same camera as the one presented in Gerdes et al., 2020 [5], the LocCam, is chosen for the development of the obstacle detection system proposed in this project.

Regarding the type of algorithms that will be used, computer vision-based autonomous navigation is a mature and evolving subject. In more recent years, accuracy in several computer vision subjects such as object detection, classification, and segmentation have been improved with the help of Deep Learning models like Convolutional Neural Networks [12]. However, machine learning still significantly lacks interpretability and determinism despite its impressive results. The algorithms used for space missions are conservative since a single mistake can lead to the crash of the rover and the premature end of a mission that has been planned for many years. Therefore, it is commonly favored to use deterministic algorithms such as "classical" computer vision. Moreover, Deep Learning models often require high computational power or specific processors such as Graphics Processing Units (GPU) which, similarly to LiDars, require a significant amount of power and have not been fully developed for space applications yet.

The development of the model will therefore be based on a classical computer vision approach with data obtained from a stereo camera and the pose obtained via localization as used in the *Efficient Navigation* system.

2. MODEL OVERVIEW

The system assumes a pair of images from a stereo camera and the current rover pose as input. The obstacle

detection will then perform the following list of image processing and obstacle classification methods in order to detect obstacles:

1. Segmentation
2. Stereo Matching and Height Extraction
3. Obstacle Classification

The location of those detected obstacles is the output of the system.

2.1. Segmentation

The model performs the initial detection on the left stereo image via segmentation to isolate regions of interest for the subsequent stereo matching operations. The main goal is to reduce the amount of computation done later, and therefore the segmentation aims at extracting only the regions where candidate obstacles are on the image.
Image Segmentation is the process of partitioning an image into distinct regions. It is a matured Image Processing technique that has significantly evolved over the years, and many methods exist nowadays, such as Thresholding, Clustering, and Region Growing. The simplest and fastest case of Image Segmentation is Thresholding. Its most basic usage outputs a binary image where each pixel is classified depending on the "lesser than" boolean comparison between the pixel's intensity value $I(u, v)$ and a given threshold t.
The main issue behind thresholding is that it has no way of being robust to illumination changes as the same threshold is given globally for all pixels in the frame. In adaptive thresholding, the intensity value of a pixel is compared not with a global threshold but with regard to the values of its neighboring pixels. Whereas global thresholding only uses a threshold, in the adaptive method, both the $BlockSize$, i.e., the size of the neighboring square in which pixels are considered, and the threshold t need to be considered.
The method used for this model is with a Gaussian-weighted average of the neighboring pixels, which implies a larger weight for pixels that are closer to the pixel of interest (u, v). The method initially computes a Gaussian-weighted average $\hat{\mu}(u, v)$, and thresholds using the following equation:

$$f(u,v) = \begin{cases} 0, & \text{if } I(u,v) < \hat{\mu}(u,v) - t \\ 1, & \text{otherwise} \end{cases} \quad (1)$$

Another thing to bear in mind is that the model considers the pixel's intensity, as mentioned above. However, the images have three channels, i.e., RGB. Therefore, one way would be to perform the adaptive thresholding separately on each channel, but the classification would then be on red, green, and blue separately. Not only would it not take the full color into account, but more specific information can be extracted from the images. Another way is to transform the initial image from RGB to HSV, which more closely align with the way human vision perceives color-making attributes.
The overall operations to segment the left stereo image are as explained in the following:

1. RGB to HSV transform.
2. Generate empty image I_final of same size as input image with all pixels set to 0.
3. For each channel we intend to segment:
   (a) Perform Median Blurring to delete noise.
   (b) Morphological Closing to link similar neighbouring regions.
   (c) Morphological Opening to delete remaining noise.
   (d) Adaptive thresholding of corresponding $BlockSize$ and threshold t for the given channel.
   (e) Activation of all "1" labeled pixels in I_final
4. Morphological Opening on I_final to delete noise.
5. Morphological Dilation on I_final to expand the segmented area to capture borders of obstacles.
6. Get the contours and the Bounding Boxes of each individually segmented area.

The final result of the segmentation process can be seen in Figure 2.

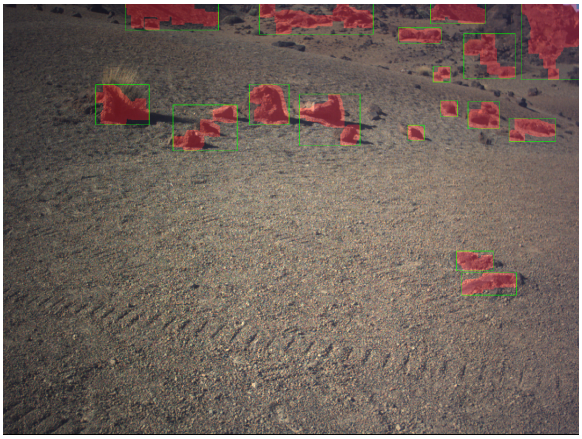


Figure 2. Original image with drawn contours and Bounding Boxes

2.2. Stereo Matching and Height Extraction

The model produced a list of candidate obstacles through segmentation and now can move on to the successive step. In order to gain more knowledge about the given candidate obstacles, this process intends to extract 3D information using the following three subprocesses: Obstacle Stereo Matching, Feature Extraction and Matching, and Height Extraction
The desired output of this process is a list of candidate obstacles' locations and their corresponding heights and confidence scores.

2.2.1. Obstacle Stereo Matching

Once again, the goal of the model is to avoid useless computation. In order to do so, this process aims at finding regions where matching features would be located on the right image using knowledge of the location of the

candidate obstacles on the left image. Indeed, matching features from a small region in the left image with features extracted over the entire right image would result in a large amount of computation to compare non-corresponding features. Therefore, the idea behind it is to use the bounding boxes of the left image obstacles and locate their right image counterparts. The technique used for this model is a specific type of template matching.
Considering that the objects to match do not go under large homomorphic transformations from left to right stereo images, that for each object on the left image, the matching object on the right image will be shifted to the left, and assuming little stereo image rectification a specific Template Matching technique customized for the given problem is used on a cropped part of the right stereo image, as can be seen in 3.
Considering T as the template to match, I the test image, (u, v) the pixel of interest on I where the similarity is measured, (u', v') varying from $(0, 0)$ to $(width, height)$ of the template and the similarity measure used being the Normalized Correlation Coefficient, the similarity for each (u, v) coordinate on I goes as follows:

$$R(u,v) = \frac{\sum_{u',v'}(T'(u',v')\cdot I'(u+u',v+v'))}{\sqrt{\sum_{u',v'} T'(u',v')^2 \cdot \sum_{u',v'} I'(u+u',v+v')^2}} \tag{2}$$

2.2.2. Feature Extraction and Matching

Now that we have a set of corresponding bounding boxes between left and right images, some more precise stereo corresponding information can be extracted through the use of Features. Features are pieces of information linking a position on an image having specific properties. They usually describe edges, points, or objects such as the templates used before. In this model, three types of features operations will be used: Feature Detection, Feature Extraction, and Feature Matching
There exist multiple ways of detecting features, such as Corner Detection, Edge Detection, Blob Detection, and Ridge Detection. Whereas Corner, Edge, and Ridge detections aim at finding sudden changes in the image's color, intensity. The type of Feature Detection that was picked for this model is a Corner Detection method. More precisely, it is the Features from Accelerated Segment Test (FAST) [10] algorithm. The main advantage of FAST is the reason itself of its name; it is computationally very efficient and allows for quick detection of numerous features.
For each candidate obstacle, features are then detected for the left and right regions where they appear.
Where the FAST algorithm strength comes in computational efficiency, it lacks complexity in describing a feature and will later conclude to a poor feature matching process. One of the most commonly used feature descriptors is the Scale-Invariant Feature Transform (SIFT) [8].
Therefore, all features detected with the FAST detector are then described using the SIFT descriptor for later comparison.
Feature Matching relates to linking corresponding features from different images. After going through the feature detection and feature extraction processes, the system is given corresponding sets of features descriptors for each candidate obstacle in both left and right images. Therefore, this process aims at finding pairs of matching features for each candidate obstacle. The ultimate reason is that 3D information regarding the obstacles will be extracted from these pairs of features through triangulation. For each candidate obstacle, the matching algorithm will try to link each left image feature to a feature on the right image's corresponding obstacle through the use of a search distance algorithm. The similarity measure used varies on the type of descriptor. For this model, the use of a Brute-Force Matcher with an added ratio test as proposed by D.Lowe [8] in his paper introducing SIFT for accurate results. The goal of this ratio test is to only consider the right image's feature that matches uniquely with f_l. It assumes that the features to match have only one correspondence in the right image, which fits our use case.

2.2.3. Height Extraction

Through the help of the feature extraction and matching process, we are now given, for each candidate obstacle, a set of corresponding pixels locations on both the left and right images. In order to extract 3D information from this data, we perform triangulation. Triangulation is the process of extracting the 3D position of a point from two corresponding points on different images with the knowledge of the camera parameters and the pose of the second camera with respect to the first one which returns a set of points in the world frame for each candidate obstacle. The height and its corresponding confidence score can therefore be extracted from each set. But before computing the height, a last outlier-removal process based on the distribution of the points' depth D is used.

$$D = \sqrt{X_{lc}^2 + Y_{lc}^2 + Z_{lc}^2} \tag{3}$$

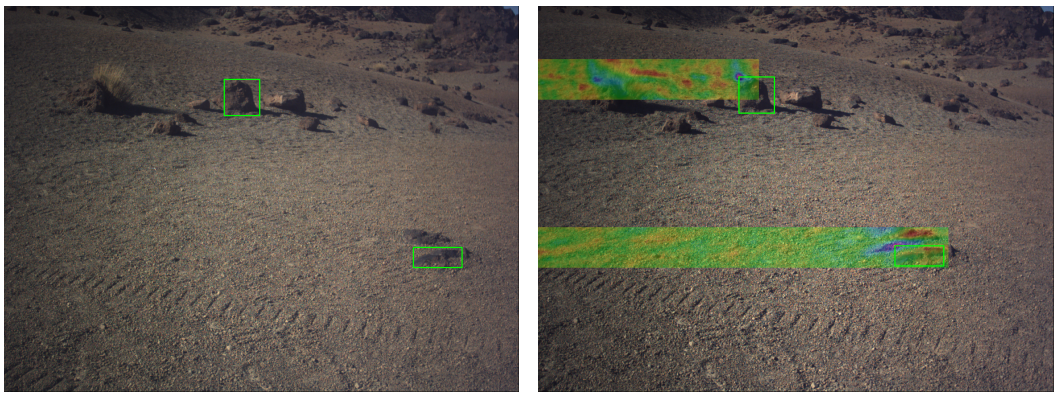
We compute the depth of all matched points for each candidate obstacle and fit a Normal Distribution by computing the mean and standard deviation of the depths. We can then remove all points that are further than a given amount of standard deviations.

The height h is then computed simply by computing the difference of Z_w values of all points in the set:

$$h = \max(Z_w) - \min(Z_w) \tag{4}$$

The confidence associated with the height depends on three factors:

1. c_{md} associated with the median depth μ_d. It relates to how far the obstacle is from the rover. Further obstacles tend to correlate with less precision in height estimation and impose less risk to the rover.
2. c_{std} associated with the depth's standard deviation σ_{std}. It relates to how spread in depth the matched features are. If the depth varies greatly, it usually implies that there are many outliers or that multiple obstacles are bundled into one candidate obstacle.
3. c_v associated with how vertically spread the matched features are in the bounding box of the candidate obstacle. As the bounding box encompasses



(a) Left image with bounding boxes covering candidate obstacles.

(b) Right image with area of interest covered by Normalized correlation score and the corresponding bounding box placed at the maximum. The opencv rainbow colormap is used. Red is low score, green is medium score, and purple is high score.

Figure 3. Obstacle Matching Process

the candidate obstacle from top to bottom, if features are only matched in the lower half, for example, the extracted height will be lower than what it truly is.

The overall confidence c_h is the harmonic mean of the three confidences stated above:

$$c_h = \left(\frac{c_{md}^{-1} + c_{std}^{-1} + c_v^{-1}}{3} \right)^{-1} \tag{5}$$

2.3. Obstacle Classification

The segmentation and height extraction processes have supplied the model with a list of candidate obstacles with corresponding locations and confidence scores. The obstacle classification process's goal will be to determine if these candidate obstacles are actual obstacles or not. Contrary to the previous processes that independently detect candidate obstacles from previous images, the obstacle classification will consider previous candidate obstacles and newly detected ones to keep a memory of the detections. The method used to keep track of the detections is called Bayesian Inference. It uses Bayes' theorem to iteratively update the probability of a hypothesis as more data from the sensors becomes available.

In our case, the hypothesis is the occurrence of an obstacle at a given location. To be able to separate spatial elements as regions, the area around the rover is discretized as a grid-like 2D occupancy local map with each cell representing a $10 \times 10\ cm^2$ region. Please note that the following equations have been retrieved from the Robot Mapping course of the Albert-Ludwigs-Universität Freiburg [6]. We consider the probability distribution of the map m as the product of the probabilites over the cells m_i. We can estimate the map given sensor data $z_{1:t}$ (i.e., the obstacle detection algorithm) as well as the pose of the rover $x_{1:t}$ over multiple detections from the initial to the t^{th} timestamps.

$$p(m|z_{1:t}, x_{1:t}) = \prod_i p(m_i|z_{1:t}, x_{1:t}) \tag{6}$$

Where $p(m_i|z_{1:t}, x_{1:t})$ can be rewritten using Bayes' Rule and Markovian properties. The ratio of the probability of a cell being occupied divided by the probability of a cell being not occupied results in:

$$\frac{p(m_i|z_{1:t},x_{1:t})}{p(m_i^c|z_{1:t},x_{1:t})} = \frac{p(m_i|z_t,x_t)p(m_i|z_{1:t-1},x_{1:t-1})p(m_i^c)}{p(m_i^c|z_t,x_t)p(m_i^c|z_{1:t-1},x_{1:t-1})p(m_i)}$$
$$= \frac{p(m_i|z_t,x_t)}{1-p(m_i|z_t,x_t)} \frac{p(m_i|z_{1:t-1},x_{1:t-1})}{1-p(m_i|z_{1:t-1},x_{1:t-1})} \frac{1-p(m_i)}{p(m_i)} \tag{7}$$

The main issue arising from such a formula is the computational demand of performing these operations. A way of reducing the complexity is to switch to the Log Odds Notation. Where we actually use the logarithmic value of the ratio:

$$l(m_i|z_{1:t}, x_{1:t}) = \log \left(\frac{p(m_i|z_{1:t}, x_{1:t})}{1 - p(m_i|z_{1:t}, x_{1:t})} \right) \tag{8}$$

And the ratio in Eq. 7 becomes a sum:

$$l(m_i|z_{1:t}, x_{1:t}) = l(m_i|z_t, x_t) + l(m_i|z_{1:t-1}, x_{1:t-1}) - l(m_i) \tag{9}$$

Considering the initial probability of a cell being occupied to be unknown: $p(m_i) = 0.5$, the sensor model $p(m_i|z_t, x_t)$ is computed depending on the height confidence c_{hi} for cell i:

1. If an obstacle has been detected on cell i and has a greater height than a given threshold:

$$p(m_i|z_t, x_t) = \frac{1 + c_{hi}}{2} \tag{10}$$

2. If an obstacle has been detected on cell i and has a lower height than a given threshold:

$$p(m_i|z_t, x_t) = \frac{1 - c_{hi}}{2} \tag{11}$$

3. If no obstacle has been detected on cell i:

$$p(m_i|z_t, x_t) = c_{nd} \tag{12}$$

With a chosen constant $c_{nd} \in [0, 0.5]$. Implying the confidence that there are no obstacles given that cell i is in the field of view.

Furthermore, it is essential to note that only the cells in what is considered to be the field of view of the stereo camera are updated.
We have yet to identify what is an obstacle. First of all, from the set of heights and their corresponding confidences, we consider a region to have a potential obstacle if its height is greater than a given threshold for the rover used, making driving over such a height detrimental for a safe traverse. Finally, if the height is indeed greater than the threshold, Eq. 10 is used to determine $p(m_i|z_t, x_t)$. If the detected obstacle has a lower height than the threshold, Eq. 11 is used. However, if nothing has been detected on a pixel in the field of view, Eq. 12 is used to determine $p(m_i|z_t, x_t)$ which is finally converted to $l(m_i|z_t, x_t)$ using Eq. 8 and denoted $detection_i(t)$ for cell i and step t.
The update of the occupancy map with value $l_i(t)$ for step t is the following:

$$l_i(t) = detection_i(t) + l_i(t-1) - l(m_i) \qquad (13)$$

An example can be seen in Figure 4. The confidence map can be seen to be zero for all regions where no detection happens. The occupancy map evolves in a way that recurring obstacles get a higher score whereas the region inside the field of view, drawn as a triangle with a $10\,\mathrm{m}$ height, gets a lower score if no detection has happened. Finally, a cell is classified as an obstacle when its value on the occupancy map exceeds a given threshold. This creates the Obstacle Map being the final classification of an obstacle. However, obstacles can be deleted from the obstacle map if the corresponding occupancy map cell value gets lower than the same threshold. This allows for deleting false positives.

3. RESULTS

The model was tested on two different datasets and was afterwards implemented on a rover at the Planetary Robotics Laboratory (PRL) - European Space Agency (ESA).

3.1. Tenerife Dataset

The Tenerife Dataset was acquired in June 2017 at Mount Teide on Tenerife in the Canary Islands, Spain. Mount Teide is a volcano that can be considered as a Mars-like terrain. Therefore, the dataset aims to resemble a planetary exploration mission on Mars and can be used to test the proposed model in the correct type of situation. The dataset was recorded with the Heavy Duty Planetary Rover (HDPR) Boukas et al., 2016 [4]. The Digital Elevation Model (DEM) is used to create a ground truth of the obstacles positions and heights and the LocCam's images (FLIR Bumblebee 2 stereo) as well as the pose will be used as inputs to the system. Examples of the stereo images were used in the previous figures (2, 3, 4) and the resulting detections can be seen in Figure 5.
From Figure 5, one can observe that all ground truth obstacles inside the propagated field of view have been detected with an additional false positive corresponding to a rock smaller than the height threshold. The $f_{1.5}score$ (F-beta measure with $\beta = 1.5$) associated with the global object-wise detection is 0.95.

3.2. Katwijk Beach Dataset

The Katwijk Beach Dataset, 2016 [7], is a field test that was conducted on a beach in the area of Katwijk. This dataset comprises of two traverses with different sparsity of obstacles. The obstacles are of known sizes (cardboard made) and their positions are known. This dataset offers different challenges than the Tenerife one. The recorded images suffer more significant illumination artifacts and make up for a more difficult stereo matching process. Moreover, even if the rocks are simpler in shape and usually larger, they lack a strong contrast with the background that could be observed on Mount Teide as seen in Figure 6.
An additional detail to take into account is that the pose of the rover has been approximated. Indeed the absolute position is determined by two sets of Trimble GNSS Antenna&Receiver pairs, but the orientation is not given and was therefore approximated using consecutive position points. The resulting detections can be seen in Figure 7.
Even though the images are quite challenging for the model and it can be seen that the localization of the obstacles is not as accurate as for the Tenerife case, only one out of fourteen obstacles has been missed, and there are two false positives which represent a $f_{1.5}score = 0.91$ object-wise. However, several detections do not lie on the ground-truth obstacle which is mainly caused by two factors. The first being the approximated pose, the second being the difficulty of performing accurate stereo matching on corrupted images as seen in Figure 6.

3.3. Planetary Robotics Laboratory

Finally, after testing on two different datasets, the model was ultimately implemented on a rover and tested on the Mars Yard of PRL-ESA. The rover used is the Martian Rover Testbed for Autonomy (MaRTA) and was designed to mimic a half-scaled ExoMars rover. It drives at slow speeds below $10\,\mathrm{cm/s}$ and is equipped with stereo cameras. The entire motion planning system is implemented using ROS2 navigation stack Nav2 [9]. The stereo camera used is a FLIR Bumblebee 2 stereo for which a ROS2 driver was developed. The onboard computer is a PICO511LG-i7-7600U equipped with an Intel Core i7 processor. The pose of the rover is given using a Vicon motion capture system. The rover and the testing situation can be seen in Figure 8.
Figure 9 shows the results of testing the rover on the Mars Yard. A costmap is generated and updated as the model detects new obstacles. In turn, the rover updates its path to avoid the newly detected obstacles. Moreover, the obstacles were placed to create narrow corridors to test the precision of the detected location. Therefore, an added $\sim 20\,\mathrm{cm}$ of detected distance with regards to the ground truth would imply a crash. Finally, the rover was able to safely perform the traverse under the narrow corridors conditions. It dynamically updated the costmap at a frequency of $1\,\mathrm{Hz}$ and allowed the rover to avoid all obstacles places on the Mars Yard.

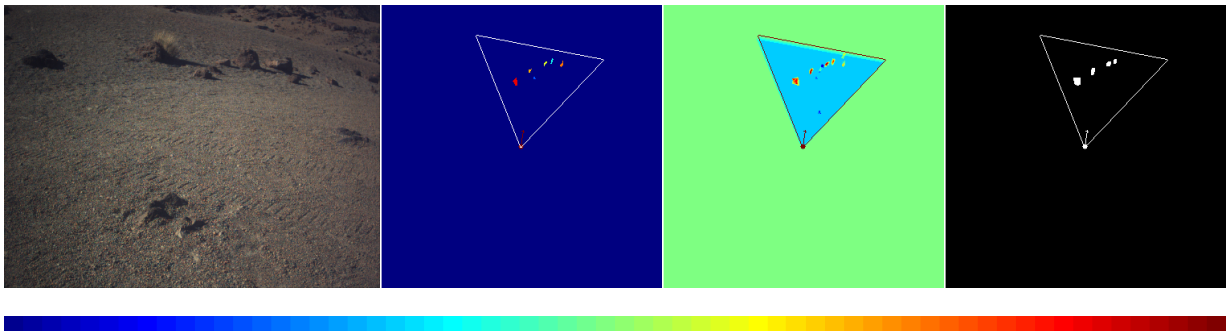


Figure 4. Example of Obstacle Classification. From top to bottom: the rover goes forward toward the rocks. From left to right: Left image, Confidence Scores, Occupancy Grid, Obstacles The rover is drawn as a circle with an arrow indicating its direction. The colormap is shown at the end with low score as blue and high score as red.

Figure 5. Tenerife final tuning results. Blue is propagated field of view, red is ground truth obstacles and green is detected obstacles.

Figure 6. Example of left stereo images for the Katwijk Beach dataset.

4. CONCLUSION

This paper describes the development of a visual obstacle detector for planetary exploration. The goal was to perform detections at a medium to close range to avoid obstacles while still enabling smooth trajectories. Another key requirement was for the model to perform real-time taking into account the driving speed of planetary rovers. The results obtained from the Tenerife dataset showed that the detection range could go up to $20\,\mathrm{m}$. For images affected by blurring and artifacts, however, the range was lowered down to $8.5\,\mathrm{m}$ but still proved to be much larger than the $0.5\,\mathrm{m}$ range from the initial benchmark model [5]. On both datasets, considering the pose uncertainty, the model resulted in $f_{1.5}$ scores over 0.9 for detection of obstacles over the entire traverses.

The integration of the model on the MaRTA rover proved to perform accurate obstacle detection and localization in real-world scenarios to navigate through narrow corridors on the PRL's Mars Yard. It also showed the real-time running of the algorithm at $1\,\mathrm{Hz}$, which is considered real-time taking into account the maximum velocity of the rovers at the PRL going up to $10\,\mathrm{cm/s}$. It succeeded as a final Proof of Concept on the feasibility and performance of the proposed obstacle detector.

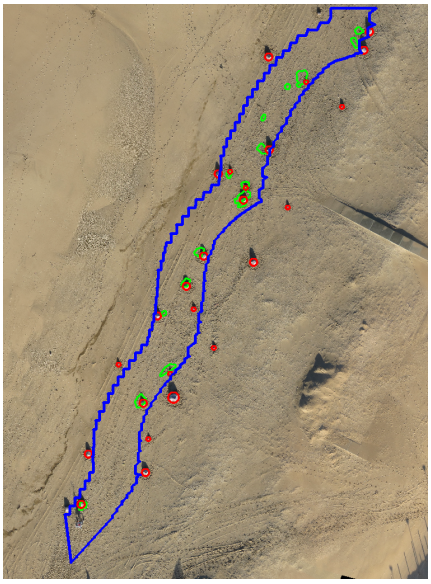
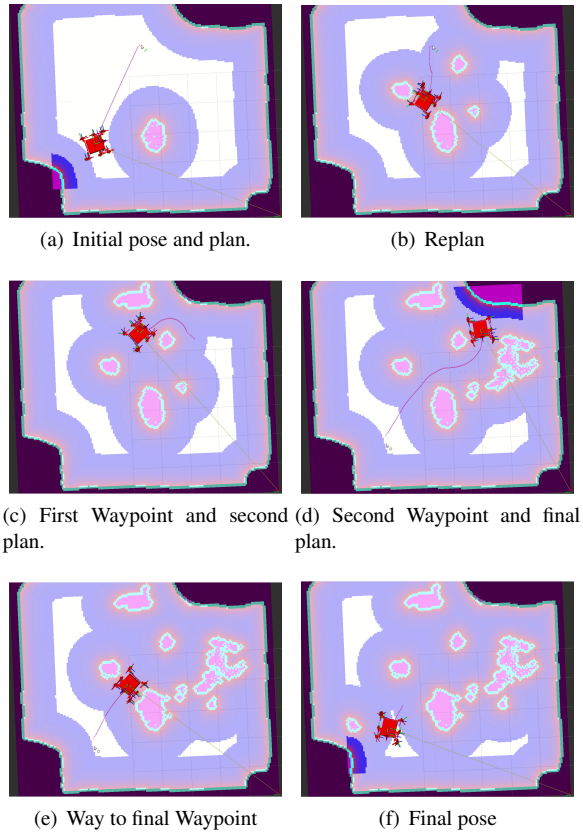


Figure 7. Katwijk Beach final tuning results. Blue is propagated field of view, red is ground truth obstacles and green is detected obstacles.

Figure 8. View of the test setup with the MaRTA rover at the initial pose.



(a) Initial pose and plan. (b) Replan

(c) First Waypoint and second plan. (d) Second Waypoint and final plan.

(e) Way to final Waypoint (f) Final pose

Figure 9. Nav2 Costmap as the rover autonomously navigates through the Mars Yard.

Overall, the proposed model has shown to meet the given requirements, with a detection range exceeding 10 m and running real-time on a test rover at 1 Hz with a high accuracy. As such, it offers an extension to the PRL's existing short range hazard detector and can be used either on its own or in conjunction with it.

ACKNOWLEDGMENTS

Thanks to the Automation & Robotics Section of the ESA for their guidance and support in conducting this research.

REFERENCES

[1] The Mars 2020 Rover's "Brains". https://mars.nasa.gov/mars2020/spacecraft/rover/brains/. [Online; accessed 01-April-2022].

[2] J. Andersson, D. Hellstrom, S. Habinc, R. Weigand, and L. Fossati. Current and Next Generation LEON System-On-Chip Architectures for Space. Workshops on Spacecraft Flight Software, 2012.

[3] M. Azkarate, L. Gerdes, L. Joudrier, and C. J. Pérez-del-Pulgar. A GNC architecture for planetary rovers with autonomous navigation. In 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 3003–3009, 2020.

[4] E. Boukas, R. Hewitt, M. Pagnamenta, R. Nelen, and M. Azkarate. HDPR: A mobile testbed for current and future rover technologies. 06 2016.

[5] L. Gerdes, M. Azkarate, J. R. Sánchez-Ibáñez, L. Joudrier, and C. J. Perez-del Pulgar. Efficient autonomous navigation for planetary rovers with limited resources. Journal of Field Robotics, 37(7): 1153–1170, 2020. doi: 10.1002/rob.21981. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21981.

[6] W. B. Gian Diego Tipaldi. Lecture notes in robot mapping. http://ais.informatik.uni-freiburg.de/teaching/ws17/mapping/pdf/slam10-gridmaps.pdf, 2014.

[7] Hewitt and e. a. Boukas. The katwijk beach planetary rover dataset. International Journal of Robotics Research, Manuscript IJR-10-1177, 2016.

[8] D. Lowe. Object recognition from local scale-invariant features. In Proceedings of the Seventh IEEE International Conference on Computer Vision, volume 2, pages 1150–1157 vol.2, 1999. doi: 10.1109/ICCV.1999.790410.

[9] S. Macenski, F. Martin, R. White, and J. Ginés Clavero. The marathon 2: A navigation system. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020.

[10] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, volume 2, pages 1508–1515 Vol. 2, 2005. doi: 10.1109/ICCV.2005.104.

[11] D. Townson, M. Woods, and S. Carnochan. ExoMars VisLoc - The industrialised, visual localisation system for the ExoMars rover. 06 2018.

[12] Z. Zhao, P. Zheng, S. Xu, and X. Wu. Object detection with deep learning: A review. CoRR, abs/1807.05511, 2018. URL http://arxiv.org/abs/1807.05511.