Diplôme Ingénieur ISAE-SUPAERO

INTERNSHIP REPORT

Skyline matching: absolute localisation for planetary exploration rovers

VILLANUEVA ROURERA Emma

Company tutor: LE CABEC Loïc

Referent professor: PÉRENNOU Tanguy

Internship period: 04/04/2022 - 31/09/2022

Date of submission: 06/09/2022

Acknowledgements

Foremost, I would like to express my deepest gratitude to Loïc LE CABEC, for making this experience possible. He eased my way into the project by introducing me to its different parts, he guided me through all the internship, he was always there to help me overcome the difficulties and he listened and took my opinion into account at all times.

I would like to continue by thanking my project leader, Thierry Germa, for welcoming me into the team and making me feel integrated from the very start. He was quite busy conducting many different projects, but he was always available to listen to my needs and give me some valuable advice. I am especially grateful to him for giving me the opportunity to participate in a scouting trip in Bardenas, Spain, to perform field acquisitions with the Magellium robot and generate data for my internship.

My sincere thanks go to the architect of the project, Vincent DELORT. He showed me the different technologies used in the project and the good practices to perform. He also helped me solve several technical problems that I encountered when starting the implementation in C++, and he did it with good grace and making sure I understood the root cause and the right steps to take.

I would like to thank Philémon FIESCHI, another team member. He showed great interest in my work and gave me good feedback during several project meetings. He contributed to the great ambience within the team.

I would like to thank Thomas RISTORCELLI, the leader of the Imagery and Applications unit, for checking on the progress of my internship and showing himself available to listen to my needs.

I express my gratitude to all the Magellium workers for their kindness, availability and willingness to help. Without them, this experience would not have been as remarkable. Special thanks go to the other interns, who helped create a great ambience at work.

Finally, I am extremely grateful to my parents and my sister, for guiding me and supporting me through all my decisions. Their trust and unconditional help have given me the confidence I needed to become the person I am today.

Table of contents

GLOSSARY

ALPER: Absolute Localisation for Planetary Exploration Rovers

CNIG: *Centro Nacional de Información Geográfica*

CNES: *Centre national d'études spatiales*

CNN: Convolutional Neural Network

DEM: Digital Elevation Model

DP: Dynamical Programming

ESA: European Space Agency

ETRS89: European Terrestrial Reference System 1989

FCN: Fully Convolutional Network

FoV: Field of View

GDAL: Geospatial Data Abstraction Library

GIS: Geographic information system

GPS: Global Positioning System

HOG: Histogram of Oriented Gradients

IGN: *Institut national de l'information géographique et forestière*

ISO: International Organization for Standardization

OpenCV: Open Computer Vision

PDD: Preliminary Design Document

QGIS: Quantum Geographic Information System

ROI: Region Of Interest

SIFT: Scale-Invariant Feature Transform

SVM: Support Vector Machine

UAV: Unmanned Aerial Vehicle

1 INTRODUCTION

Absolute localisation in planetary exploration missions is crucial to allow navigating towards a certain target, deposing or collecting georeferenced samples or path planning with obstacle. In current missions, dead-reckoning techniques like inertial navigation, wheel odometry and Visual Odometry (VO) are used to update the position of the rover during its traverse. However, these relative localisation techniques present an error drift that grows with the distance travelled. Thus, absolute measurements are necessary to recalibrate the position and orientation every few meters or to estimate the position when no prior calculation is available. Furthermore, the communication delay makes it particularly difficult for effective scientific data exchange during planetary missions, so autonomous navigation becomes crucial for an effective long-range exploration.

The ALPER project is part of Magellium's Imagery and Applications unit and falls within the domain of space robotics. It seeks to explore absolute localisation techniques for rovers on planetary missions, particularly on Mars. It explores solutions like Tie Points Tracking, Constellation Matching and Dense Image Co-registration. These techniques are based on comparing rover acquisitions with orbital images and they all seek to correct available position and orientation estimates obtained through dead-reckoning techniques. The two first approaches (Tie-Points Tracking and Constellation Matching) fall into the "interest point matching" category since they are based on finding correspondences between visual/3D landmarks. The third approach belongs to the "dense image/terrain matching" category where all the radiometric information of the rover images is used during the matching process with the orbital images. The first approach relies on constant operator intervention, while the second one depends on the presence of particular features along the rover's path and needs manual initialisation once per sol[1]. The third method is fully autonomous but less mature.

Skyline matching is a technique for absolute localisation framed in the category of autonomous long-range exploration. This is an area of active research, as it becomes crucial to recalibrate position during long traverses (updating problem) or to estimate position with no *a-priori* information when external measures are unavailable (drop-off problem). This is the case of planetary exploration, where no GPS infrastructure exists and the magnetic field is practically non-existent, not allowing for compass measurements. Skyline matching is also used for port security, shipment location or geo-location, as well as for augmented reality applications.

The skyline can be defined as the boundary that separates the sky and non-sky regions (urban areas, mountains, forests or seas). It is often used as the main source of information for geo-location and navigation purposes as it is the most notorious and informative feature, especially for natural scenarios, and it is particularly stable across time, climate and seasonal changes.

A skyline matching algorithm is based on comparing the skyline from an orbital image (usually rendered from a Digital Elevation Map) and the skyline extracted from the rover's imagery. It consists of several distinguished parts: skyline rendering from DEM, skyline extraction, possible rectification, skyline matching and position estimation.

[1] A sol is a solar day on Mars

The main objective of my internship was to explore a solution for the lost-in-space problem, where the rover needs to estimate its position with no prior information. In the context of autonomous navigation, a different use-case was considered, where the algorithm is used in complement with the relative localisation to correct the position or orientation drift. In total, three scenarios were considered: 1) recalibrating position after landing, 2) correcting the position and orientation drift at the end of a 1 km autonomous traverse and 3) continuous execution on-board of the rover to correct the attitude drift every 50m, admitting a rough position estimate is known via relative localization methods.

I joined the Imagery and Applications Unit at Magellium, where I worked as an intern for 6 months in the space robotics team. I worked as a research and development engineer by exploring, implementing and validating a skyline matching algorithm both on real and simulated data.

In this report, we will first have an overview of the enterprise and the project team. Then, I will present my internship mission and its different work phases. We will review the state of the art on the different parts of the algorithm. We will define the different scenarios, along with the system requirements, the test parameters and the metrics to evaluate the algorithm. The general operation of the algorithm will be represented and its architecture described. We will then have a look at the implementation phase, as well as the means to obtain the data, including field tests in Bardenas, Spain. Finally, the results of the different tests will be presented and we will extract a conclusion.

2 THE COMPANY

Magellium Artal Group is a French tech company of about 250 employees specialised in geoinformation and image processing. Founded in Toulouse in 2003, the company was born to meet the needs of the French National Centre for Space Studies (CNES), the French National Institute of Geography (IGN) and Airbus Defence & Space. Since then, it has grown steadily consolidating its expertise in the geoinformation field and getting heavily involved in the development of vision-based solutions for robotics applications.

Magellium offers include consulting, technical & scientific studies, software & IT system development and software product distribution. The company has a recognized expertise in earth observation, geographic information systems, mapping technologies and vision-based systems.

The Group is structured around two main entities:

- Artal: expert in industrial software systems (embedded systems, data processing, mobility, software engineering & methods).
- Magellium: expert in imaging and geo-information (Earth observation, Geographic Information System & cartography, geo-intelligence, computer vision & robotics).

Its technical and commercial teams are located in 2 different sites: Toulouse (headquarters) and Paris (Courbevoie). Magellium is primarily involved in activities related to Defence & Security, Space, Transportation, Energy, Public and Environmental sectors. The company is ISO 9001:2008 certified on all engineering activities.

The enterprise is organized into several units according to its different domains of expertise. The Imagery and Applications (IA) unit develops vision and data processing systems in the fields of space robotics, defense and industry. Its customers are the main industrial and institutional players in these sectors (Airbus Defence and Space, CNES, ESA…).

The Absolute Localisation for Planetary Rovers (ALPER) project team is part of the IA unit and was initially formed of 5 people including the project leader, the technical leader, the code architect, the scrum master and a development engineer.

3 INTERNSHIP MISSION

The main goal of my internship was to explore existing solutions to solve the lost-in-space problem, in which the position of a rover needs to be estimated with very rough prior knowledge, for a Martian-analogue environment.

This goal can be divided into several missions or phases:

- State of the art of the existing methods based on the comparison of rover acquisitions and orbital images.
- Prototyping the identified solutions in Python or C++.
- Participating in acquisition campaigns with Magellium's robots and drones on representative terrains.
- Performance evaluation with the acquired data as well as with simulated data generated during the internship.

The solution had to be autonomous, meaning it requires no operator intervention during its standard operation. In addition, it had to be apt for implementation on-board of a rover. This meant that memory and time optimization were taken into account in the process of finding a solution. The aim was to achieve 1 km/sol of autonomous navigation.

On a personal aspect, my personal goal was to learn as much as possible about space robotics, learn to use new tools and programming languages and, more generally, learn about the workplace and agile project organization. Overall, I wished to acquire important competences for the professional world while fostering my professional integration.

4 METHODOLOGY

During my internship, I worked as a research and development engineer in the ALPER project. I worked in parallel to my other co-workers, exploring possible solutions to the lost-in-space problem involving orbital images and rover acquisitions, and finally focusing on skyline matching. The original aim was to estimate the position and orientation of a rover from a very rough prior estimation. In the end, accounting for a real usage, 3 real scenarios were considered, with different uncertainty areas and performance requirements.

The ALPER project is managed in agile, particularly in scrum. The Agile methodology involves a set of practices focused on an ongoing collaboration between autonomous and multidisciplinary teams, continuous planning, improvement and early deliveries. It encourages flexibility, to support the team's ability to respond to changes. Scrum is a framework which implements the agile methodology. It is based on an iterative method that focuses on regular deliveries, a set of roles and ceremonies. Sprints are the heartbeat of scrum. They represent timeboxed iterations of a continuous development cycle, in which particular tasks and objectives need to be accomplished. A sprint consists of at least a Sprint Planning, Daily Scrums, a Sprint Review and a Sprint Retrospective. Sprints in the ALPER project usually last 4 weeks.

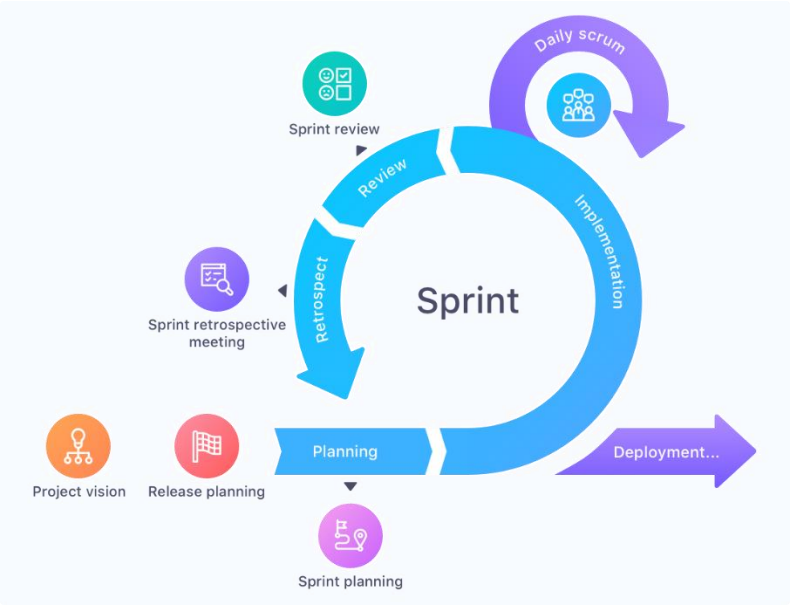


Figure 1. Scrum project life cycle [1]

As an intern, I participated in most of the project meetings, including a daily 15-minute meeting, monthly 2-hour internal reviews, reviews with the client (ESA), sprint retrospectives and sprint plannings. In contrast with the rest of the team, my work was cadenced with 2-week sprints and additional weekly meetings with my technical tutor were held to review the work done, define new objectives, and plan the different tasks to perform for the next sprint. Each sprint had different objectives and milestones to reach. A Jira Board (an agile tool) is used to keep track of all the tasks. All team members have their name on the board and can create dedicated subtasks, update their status, add information or assign them to different people. By clicking on the tasks, its information can be retrieved (description, release, topic, priority, etc.).

My internship was organised like an ESA project, with 4 distinguished phases, associated to different deliverables:

1) **System Engineering** (System Requirements Review): This phase was mainly dedicated to review the state of the art on the different existing methods to solve the lost-in-space or drop-off problem, focusing on skyline matching. Additionally, the different scenarios were defined and the system requirements were identified. The different tests to be performed were determined, with the conditions and parameters to be tested, as the well as evaluation metrics and the necessary material for the field acquisitions. The outputs of this phase were the System Requirements and the Demonstration Scenario documents.
2) **Requirements Engineering** (Preliminary Design Review): The architecture of the algorithm was defined. The different components and subcomponents composing the on-board and ground segments were described in detail. Their subfunctions, inputs and outputs were explained and represented. Two versions were defined: a basic mock-up version and an optimized final version. An activity diagram defining the usage and potential errors of the algorithm was generated. During this phase, I took Blender (a 3D computer-graphics software) in hand to test and automate the skyline rendering from a DEM through a Python script. The output of this phase was the Preliminary Design Document.
3) **Design Engineering** (Critical Design Review): The different parts of the algorithm (panorama assembly, skyline detection, skyline transformation and pose estimation) were coded in C++. A demo executable was created and unitary tests were performed for all of the components, as well as for the whole function. Simulated data was generated using Blender and QGIS (GIS software) was used to prepare the DEM. The outputs of this phase were technical notes describing the modifications made with respect to the Preliminary Design Document (PDD) and the future improvements.
4) **Validation and Acceptance** (Qualification/Acceptance Review): The algorithm was tested using real and simulated data, the results were analysed and potential improvements were identified.

5 STATE OF THE ART

In this section, the state of the art on the main parts of the skyline matching algorithm is reviewed, and the method chosen is justified. The work present in this section is a summarized version of the technology review document submitted during my internship.

5.1 SKYLINE RENDERING

The first step for the skyline matching algorithm is to render skylines from the DEM. That is, for every position in a 2D sampling grid, get the panoramic horizon line that would be seen according to the rover's camera characteristics and store it in a database along with its corresponding position.

In the literature, not much focus is put on this crucial step on the algorithm. Stein and Medioni introduced in [2] the key idea of pre-compiling the horizons and their line segments off-line according to their chosen skyline representation. This allows to speed up performance during on-board execution.

Some authors [3], [4], [5], [6] mention the use of a ray-tracing algorithm to extract the elevation profile for every point in the grid. Babaoud et al. [7] talk about using ray-casting to render the silhouettes into a 2D cylindrical image. Tzeng et al. [6] and Pan et al. [8] propose an approach in which the full scene is rendered at each sample point via overlapping images.

A different approach is taken in [10] by Rodin et al., who use the camera model to project the skyline elements from the world coordinates to the image coordinates. Baatz et al. [11] render a cubemap of the textureless DEM for every grid position and they extract the horizon by checking the rendered sky colour. Several authors mention the use of OpenGL [12] and depth map extraction [13]. Nefian et al. [12] talk about using a low coverage high-resolution DEM augmented by a high coverage low resolution DEM to satisfy both the wide-coverage requirements for horizon rendering and the high-resolution requirements for terrain matching while accommodating the memory constraints of a typical GPU.

In our algorithm, we reuse the idea of [10] of using a textureless DEM and image segmentation by colour and the idea of using several sizes and resolutions [12]. This idea is simple to implement and the required capacity is available at Magellium's site.

5.2 SKYLINE EXTRACTION

Extracting the skyline from an image has been a topic of interest for several years, as it plays a decisive role in vision-based navigation for UAVs or planetary rovers, mountainous geo-localisation, port security and augmented reality applications. It consists in extracting the skyline from an image.

Related work on this area shows two classic approaches for solving this problem: region-based or contour-based methods. Region-based solutions make use of image segmentation algorithms in order to distinguish the sky from mountainous or urban areas. In edge-based approaches, the characteristics of skylines are defined to select from candidates in an edge map.

Traditional region-based approaches perform a vertical-line search based on intensity thresholding. For every column starting from the top of the image, the first pixel whose intensity is below a certain threshold is selected as a skyline pixel [14], [3]. Both of these solutions have the downside of not being robust to complex environments.

Lie et al. [15] set the base for edge-based methods with their Dynamic Programming approach. They construct a multi-stage graph from an edge map. Links and costs are set based on adjacencies and vertical pixel location, and a shortest path algorithm is used to find the minimal cost solution between two virtual nodes. A gap filling step is performed using high-cost dummy nodes to allow for a certain tolerance gap. Similar approaches can be found in [16], [17] and [18]. Woo et al. [19] use global energy minimization for all available edge-pixel paths via contrast cost and homogeneity cost. Other approaches [20] include hierarchical methods, in which non-horizon pixels are excluded step by step.

More recent works explore the use of Machine Learning to segment the image. As machine learning is not used for space applications yet, our first choice is to use is to use traditional methods. However, a quick review of these more recent methods has been made to check for key ideas in case results obtained with traditional methods were not satisfactory.

These approaches can be classified into supervised learning approaches and deep learning approaches. The first ones use the extraction of explicit feature descriptors and classification or direct discrimination based on pixel intensity [21], [22], [23], [24] while the latter are mainly based on CNN or on fine-tuning existing general scene parsing deep networks [25], [26], [27], [28].

A focus is made on Ahmad et al.'s work in [22]. They use SVM and CNN classifiers trained with normalized pixel intensities, they obtain a dense classification score map according to the likelihood of the pixels belonging to the horizon line (horizon-ness) and they apply Dijktra's shortest path algorithm to find the horizon on a multi-stage graph. In [24], they additionally boost the score of edge pixels.

For the sake of robustness, we adapt the key idea of [22] and [24] of using a dense multi-stage graph with the edge-ness inside the cost function. However, we redefine the horizon-ness on the basis of their edge-based method [29] and other traditional edge-based methods.

5.3 SKYLINE RECTIFICATION

Transforming the skyline is a crucial step to allow matching skylines with different roll, pitch or yaw angles. This step is not usually taken into account in the literature, as the camera is assumed to be levelled [32], [3], or the angle small enough to not affect the algorithm [11]. Other approaches, like [21], search for a skyline representation that is robust to rotations, by using normalized concavity features.

Some authors use vanishing point estimation in urban environments to get a pitch and roll estimate and rectify the skyline [33] or the image [34] accordingly. However, this method is based on aligning lines that are approximately upright to a common vertical orientation and thus, it is not adapted for mountainous terrain.

Several approaches can be found in the literature that try to deal with the attitude without rectifying the skyline. Some use a parameter range around the estimated pitch or roll angles in the quantized grid of

sampling points, while others include a full range of values for estimating the yaw value. This will be further explained in the skyline matching section.

In [35], Grelsson et al. use Canny edge detection and Hough voting to get a rough estimate of the camera's pitch and roll angles and warp the image accordingly. In [5], they use a CNN instead. Nevertheless, both their methods are designed to find the horizon at sea level, and are thus not applicable to planetary environments.

5.4 SKYLINE MATCHING

Skyline matching is the next key step of the algorithm. It consists in evaluating the similarity between the skyline extracted from the rover's imagery and the skylines rendered from the DEM, in order to obtain the best match and estimate the rover's position.

Research on the field of skyline matching distinguishes 2 approaches to solve this problem: feature-based or signal-based methods. Feature-based methods search for correspondences among features, like natural landmarks (peaks or depressions), while signal-based algorithms use dense structures in the image. Both techniques are highly dependent on the features and the method used to encode the skyline.

To our knowledge, Stein and Medioni [2] were the first ones to use the full skyline for their signal-based matching algorithm. They extracted super segments, which they encoded into a table, and they retrieved candidate hypothesis based on similarity.

VIPER (Visual Position Estimator for Rovers) [3] was the first algorithm specially created for planetary rovers and extensively tested on both terrestrial and lunar environments. Their signal-based approach considers Gaussian measurements and uses an evaluation function based on Bayesian statistics in order to find the best match. They store the skylines as vectors of elevation angles indexed by azimuth value.. In [32], Furgale et al. made a more robust version of the algorithm by modifying the likelihood function. A previous less accurate version of the VIPER system [36] used a feature-based approach based on peak extraction and evaluation. A similar approach can be found on Wei et al.'s work [37].

Other approaches include different encoding strategies. Several articles explore the idea of using contour words to represent the skyline within a bag-of-words approach [21], [38], that is, dividing the skyline into curvelets sampled at regular intervals. The disadvantage of this kind of approaches is its weak robustness against any rotations in the curve, which makes it inadequate for situations where pitch and roll are unknown.

In [6], Tzeng et al. use the concavity as a feature within a geometric hashing matching procedure, whereas Pan et al. [8] propose a new method for locating hilly areas using lapel points as features. In [39], Nuchter et al. reduce the skyline-matching problem to a string-matching problem. Their approaches have the advantage of being robust to the effects of scales and in-plane rotations.

Some attempts for edge matching have been made on the basis of cross-correlation on the Fourier domain [7], [5], [40]. However, these types of approaches are quite expensive computationally and are thus unsuitable for real-time applications.

In the literature, most skyline matching techniques rely on the availability of an accurate measure of the rover's attitude. However, in the present work, an estimation of the orientation is sought along with the position. As stated before, some authors have enlarged the estimation procedure by including the yaw in a 3D grid of sampling points [3], [41]. In [41], Chiodini et al. use a least square error metric for Martian rover localisation and their algorithm achieves 50-meter accuracy. Other authors include a parameter range around the estimated pitch or roll angles in the quantized grid of sampling points [38], [42].

Several approaches for correcting estimated pose and adjusting the azimuth have been studied within the context of Augmented Reality [13], [33], [43], [9] They are mainly based on cross-similarity functions.

Our approach reuses the idea of encoding the skylines as a vectors of elevation values indexed by azimuth [3] and evaluating the match with a simple least square error metric [41]. A final version of the algorithm would reuse the idea of using concavity features [6] to gain robustness to pitch and roll errors.

5.5 POSITION ESTIMATION

Pose estimation is the last key step of the algorithm. It consists in obtaining an estimate of position by applying the skyline matching metric as well as any other methods that could optimise the procedure.

Most methods found in the literature assume a rough position estimate is available (reduced search area) or they do a full-grid search [3], [37], [41]. The latter is computationally expensive but can be optimised by precompiling and storing skyline features.

Other techniques can be found, particularly on approaches conceived to estimate orientation within the context of Augmented Reality. Gupta et al. [44] use Random SAmple Grid Search (RSAGS) to select 4 random points to which apply their least sum of vertical distances metric. The same metric is then applied to all points to evaluate the best candidates. In [43], Ayadi et al. use gradient descent with a distance metric to find the best orientation estimate. Dumble and Gibbens [42] also use gradient descent to refine an initial position estimate.

Others [21], [39] use a least accurate metric for all candidates and apply ICP to find the best estimate among the resulting candidates. The disadvantage is that ICP is highly sensible to any rotations or scalings. A two-step approach is also applied in [8], consisting of a coarse matcher based on skyline features followed by a refined matcher based on Label Points.

A pyramidal approach is proposed in [5] by Grelsson et al. using the fact that the skyline does not vary notoriously within few meters of distance. A larger grid-size is used at first to find a coarse position estimate, and then the size of the grid is decreased around this position to refine the estimate.

Most of these approaches are not adapted for position estimation combined with orientation estimation, due to lack of accuracy or high computational time. However, some of these ideas could be adapted to avoid doing a full-grid search with the chosen metric and speed-up the execution of the algorithm.

6 SCENARIOS

Accounting for a real usage of the algorithm, different scenarios are considered. The scenarios were defined during a brainstorm with the client (ALPER's project team at Magellium). My objective was to verify whether skyline matching could respond to these needs.

6.1 USE-CASE DESCRIPTION

As stated, 2 different use-cases are distinguished for the implementation of the skyline matching function. The first one is the lost-in-space scenario, in which no *a-priori* information is known and position and orientation need to be estimated from a full-grid of candidates.

The second use-case is conceived on account of autonomous rover navigation and is based on a continuous approach in which the rover either refines a position estimate or estimates its orientation from a certain position estimate. This approach is used in complement with relative localisation techniques to correct their drift every few meters of traverse.

In the next sections, real scenarios for planetary missions are defined for each of these use-cases, and the system requirements for each of them are defined.

6.2 LOST IN SPACE

The lost in space scenario with no prior information is not considered to be realistic for an actual planetary mission since a rough estimation of the rover's localization will always be available in actual operational conditions. However, the closest scenario would be found **after the rover's landing**, where a non-trivial ellipse of uncertainty exists. Recent methods have allowed to reduce the size of this ellipse significantly over the last few years. As of the last Martian missions, Perseverance's landing ellipse was 7,7 by 6,6 kilometres, compared to 7 by 20 km for Curiosity.

This first approach imposes less restrictions in terms of computational time, as it should only be done once in the rover's lifetime. Additionally, accuracy constraints as not as tight as in the other scenarios as the initial uncertainty is a lot bigger, and the first obtained estimate can be further refined using other absolute localisation techniques like Tie-Points Tracking, Constellation Matching, Dense Image Co-Registration or the continuous version of the algorithm (refer to section 6.3Continuous approach).

6.3 CONTINUOUS APPROACH

The continuous approach could be used in 2 different scenarios on a planetary mission.

- **End of autonomous traverse:** In this scenario, the algorithm is used in complement with relative localisation methods to correct the position drift at the end of a journey of autonomous traverse (~1km).

- **Along the traverse:** In this scenario, the localisation function is used continuously in complement with relative localisation methods to correct the orientation drift every few meters of traverse (~50m).

This continuous approach is a lot more restrictive than the first one, as it should allow for continuous autonomous rover exploration and, consequently, its execution time is crucial to avoid accumulating errors. However, the available a priori is rather precise, which makes the task easier. Additionally, its aim is to correct an initial estimate, so accuracy is particularly important. In the last scenario, orientation is estimated but not position, so the algorithm might suffer finding the good orientation when the available position estimate is not precise enough. As the algorithm is executed along the traverse, low computational times are essential and little accuracy translates to large error accumulation.

7 SYSTEM REQUIREMENTS

7.1 FUNCTIONAL REQUIREMENTS

The functional requirements concern the system behaviour and its capabilities. They are common for all scenarios.

- **On board execution:** In the context of long-range autonomous rover exploration on planetary missions, the skyline matching algorithm shall run online, on-board of the rover, with no operator intervention.
- **Input data:** The Skyline Matching localisation function shall use ExoMars NavCam-like overlapping images and HiRISE-like orbital maps, as well as 2 GPS for obtaining ground-truth position and orientation data.
- **Physical environment:** The function shall run on a Martian-like environment, with a focus on horizon lines and impacting environmental parameters. That is, mountainous areas, including broad and irregular terrain, steep cliffs, valleys, ridges and a dusty environment.
- **Function TRL 4:** The skyline matching localisation function shall reach TR 4 at the end of the activity. TRL 4 corresponds to a model demonstrating the critical functions of the element in laboratory environment. By laboratory environment, we include a test campaign in an analogue environment, but exclude the deployment on representative target processor.
- **Measure of estimation confidence:** The skyline matching algorithm shall provide an intrinsic measure of the quality of the estimated position. This can be used to decide whether the position or attitude is updated with the new estimate, a new estimate is computed or the old one is kept.
- **Robustness:** As pitch and roll are not part of the estimation function, the function is required to be robust to small variations in these camera parameters. Particularly, the skyline matching algorithm shall be robust to 10° errors in pitch or roll angle.

7.2 PERFORMANCE REQUIREMENTS

The performance requirements define a set of criteria which stipulate how well the system completes its tasks under specific conditions. They vary depending on the scenario.

- **Size of the search area:** the size of the search area has been determined according to the uncertainty present in each scenario.
    - In accordance with recent advances on planetary missions, Perseverance landing ellipse (7.7x6.6 km) is taken as a reference and the localisation algorithm is designed to consider a search area of 7x7km for the after-landing scenario.
    - For the end-of-trajectory scenario, this zone is reduced to 40x40m to account for a 2% error in a 1km traverse. The 2% error has been estimated using Magellium's internal algorithms on representative data in the context of previous projects.
    - For the along-the-traverse situation, a search are of -10º to +10º around the current orientation estimate is considered, on the basis of a 10° uncertainty.

- **Localisation accuracy:**
    - In the first scenario, the aim is to have enough precision to be able to start the mission. The target is 5m, as the initial estimate can further be refined using the on-board version of the algorithm.
    - For the continuous approaches, the goal is to be able to fetch a sample localised on an absolute reference, so the accuracy is set to 1m for the second scenario.
    - The aim is set to 1º for the third scenario. This angular accuracy constraints the error in position to 1m considering execution every 50m.
- **Execution time:**
    - The first scenario does not impose a big constraint in terms of execution time, as the mission has not yet started. The aim is to have an estimate in less than 1 sol.
    - The second scenario is performed at the end of the journey, so the goal is to have an estimate by the start of the next journey. A 1-hour execution time is targeted.
    - The third scenario is the most restrictive in terms of execution time, as the algorithm is executed along the traverse. 5 minutes is the selected target value.
- **Spatial frequency:** due to the nature of the different scenarios, the function shall be executed with different frequencies.
    - Once in the rover's lifetime for the first scenario.
    - Once every kilometre for the second one.
    - Once every 50m for the third one. In this case, as the pose is not estimated by the algorithm, the orientation is computed from a position that is increasingly further from the ground truth, so the objective is set to constraint the error within a 2 to 5% interval.
- **RAM:** The skyline matching localisation function shall be able to run on a 256 MB RAM. This target is designed to allow for a realistic implementation taking into account rover characteristics for last planetary missions. Perseverance computers[2] for Mars 2020 are taken as a reference.
- **Flash memory:** The skyline matching function shall take less than 2 GB of flash memory. This target is adapted to the Perseverance rover used on Mars 2020 and refers particularly to the storage of the skylines rendered from orbital data and their ground truth position.

[2] https://mars.nasa.gov/mars2020/spacecraft/rover/brains/

8 TESTS

This section describes the different parameters to be tested, the evaluation metrics, the required entry datasets and the diverse means to acquire the data, both for field tests and simulated data.

8.1 TEST CHARACTERISTICS

Tests need to be performed to verify the system requirements for all 3 scenarios, as well as the general system behaviour under different circumstances. The different parts of the function need to be evaluated separately with an appropriate metric. Specifically, skyline extraction and pose/orientation estimation need to be tested. The variety of rover acquisitions needs to be defined.

8.1.1 Robustness test parameters

In this section, the different parameters possibly impacting the algorithm's performance are presented along with their nominal and degraded modes.

- **Horizon relief**
    - Nominal: the optimal relief consists of a distinct skyline with clearly identifiable and unique features.
    - Degraded: the degraded skyline would be a flat or unvarying skyline. Regular skylines contain less distinctive features for the algorithm.
- **Skyline completeness**
    - Nominal: the skyline is completely inside the camera view.
    - Degraded: the skyline is partially or totally outside the camera's field-of-view (caused by extreme topography). Extracted skyline is not complete.
- **Local relief** (camera's pitch and roll angles)
    - Nominal: rover is on a flat surface and camera is levelled (pitch and roll angles are about 0°).
    - Degraded: rover is on different types of relief (ascending and descending slopes, craters, mounds…). Pitch and roll are not 0°.
- **Presence of clutters** blurring the skyline
    - Nominal: defined boundary between sky and non-sky areas, no clutters.
    - Degraded: blurred skyline due to the presence of clutters like clouds, atmospheric dust, sandstorms…
- **Presence of elements close to the rover** (partial occlusions):
    - Nominal: no occluding elements close to the rover
    - Degraded: presence of big elements (rocks, trees, hills…) close to the rover. Close elements might occlude the distant horizon and cause any small deviations in the model to have a significant effect on the localisation result.

- **Degraded acquisition conditions** (weather, luminosity)
    - Nominal: acquisitions taken with daylight and good weather (no rain, no fog, no shadows occluding the skyline)
    - Degraded: direct sunlight, sun glare, camera over exposition, fog, darkness…

8.1.2 Evaluation metrics

For evaluating the skyline extraction, the pixel-wise absolute distance between the extracted and the ground-truth skyline is used as a metric. The ground-truth skyline will be manually extracted by an operator for the field tests.

For evaluating the position estimation, the absolute distance between the estimated position and the ground-truth GPS position is used.

For evaluating the orientation estimation, the absolute distance between the estimated orientation and the ground-truth orientation obtained with 2 GPS is used.

8.2 LIST OF CONSIDERED IMAGE SETS

Different sets of images need to be taken with to evaluate the robustness of the algorithm to the different impacting parameters independently.

Set ID	Horizon relief	Skyline completeness	Local relief	Presence of clutters	Partial occlusions	Degraded conditions
REF-SET	Nominal	Nominal	Nominal	Nominal	Nominal	Nominal
HREL-SET	Degraded	Nominal	Nominal	Nominal	Nominal	Nominal
SCOM-SET	Nominal	Degraded	Nominal	Nominal	Nominal	Nominal
LREL-SET	Nominal	Nominal	Degraded	Nominal	Nominal	Nominal
CLUT-SET	Nominal	Nominal	Nominal	Degraded	Nominal	Nominal
OCCL-SET	Nominal	Nominal	Nominal	Nominal	Degraded	Nominal
COND-SET	Nominal	Nominal	Nominal	Nominal	Nominal	Degraded

Table 1: Set of considered acquisitions

Some of these image sets are quite hard to obtain during field tests, especially those regarding the presence of clutters, degraded acquisition conditions and potentially occlusions. Thus, available datasets or simulated data will be used instead to test the robustness of the skyline matching function for some of the impacting parameters.

9 PRELIMINARY DESIGN

The Skyline Matching basic function is to estimate the pose of the rover by minimizing the errors between the extracted skyline and the georeferenced rendered skylines for a certain search area. Its nominal operation is briefly presented in this section.

The process is divided in two segments that are intended to run in two different contexts:

- **Ground segment:** its function is meant to run off-line once before running the board segment. It consists on rendering and storing the skylines from a desired search area in a DEM in the form of elevation vectors.
- **Board segment:** This component is mainly composed of the skyline extraction and pose estimation functions, added to image assembly and transformation. It is meant to run autonomously on-board of the rover until one of the following errors is raised:
    - The latest estimated pose is too uncertain after a certain number of iterations
    - Extraction failure

    In both cases, operator intervention is requested either to manually extract the skyline or to move the rover to a new position where the skyline is more distinct or appears more clearly inside the camera's Field of View. In the lost-in-space type of scenario, the function is meant to run once in the rover's lifetime. In the continuous approach, the function is supposed to run iteratively.

The preliminary version of the algorithm assumes that the rover's camera is levelled (pitch and roll are 0°) and that there are no tight constraints in terms of computational efficiency.

The overall Skyline Matching process for the continuous approach is presented on the activity diagram below.

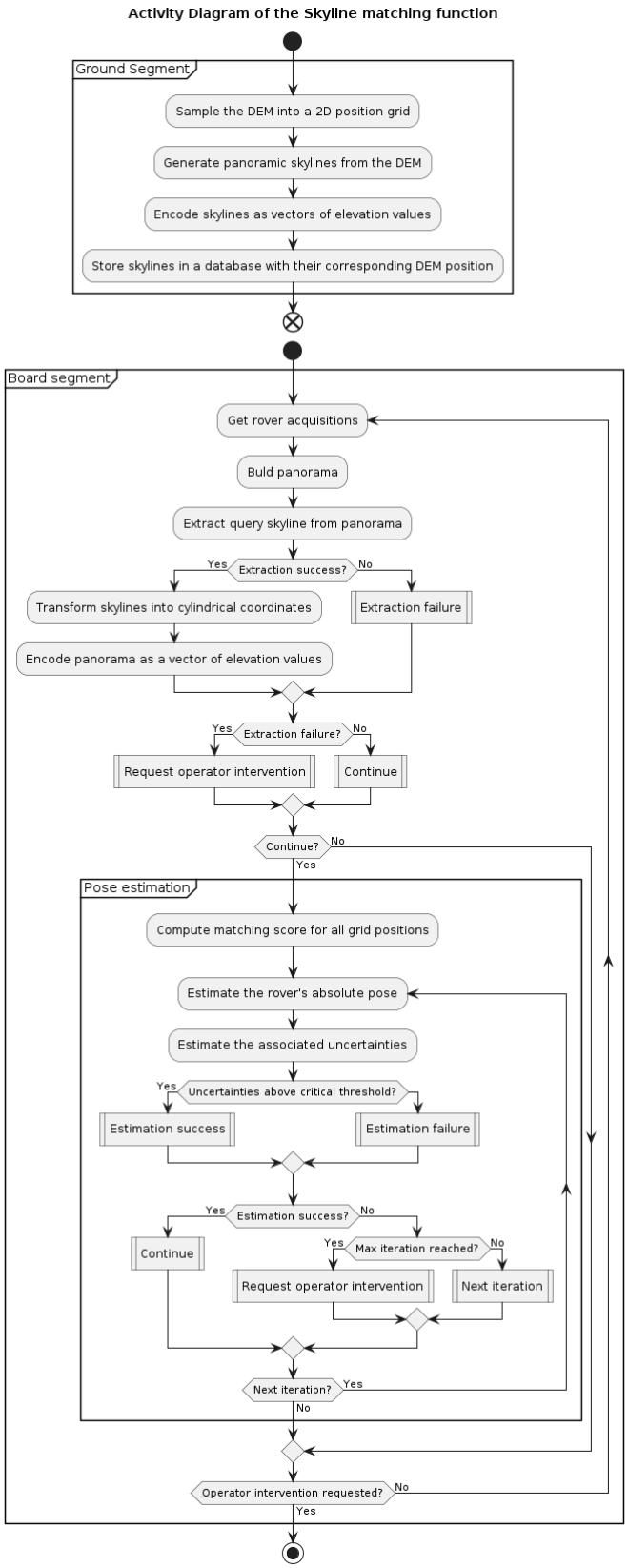


Figure 2 : Activity diagram of the skyline matching function

9.1 GENERAL ARCHITECTURE

The demonstration environment for skyline matching is composed of 3 main components:

- The **ground segment** with the user interfaces in case of unexpected behaviour and the tools for rendering the skylines from the DEM.
- The **board segment** with the core functions for extracting skylines from the rover imagery, matching them and estimating the rover's pose and orientation.
- The **demonstrator**, which manages the communication between ground and board segment, emulates the rover for the board segment and displays the product of the ground segment.

On the following pages only the board and ground segments are going to be considered as components, as testing on representative environment is out of the scope of this document.

A scheme showing all subcomponents for this mock-up version is shown in Figure 3.

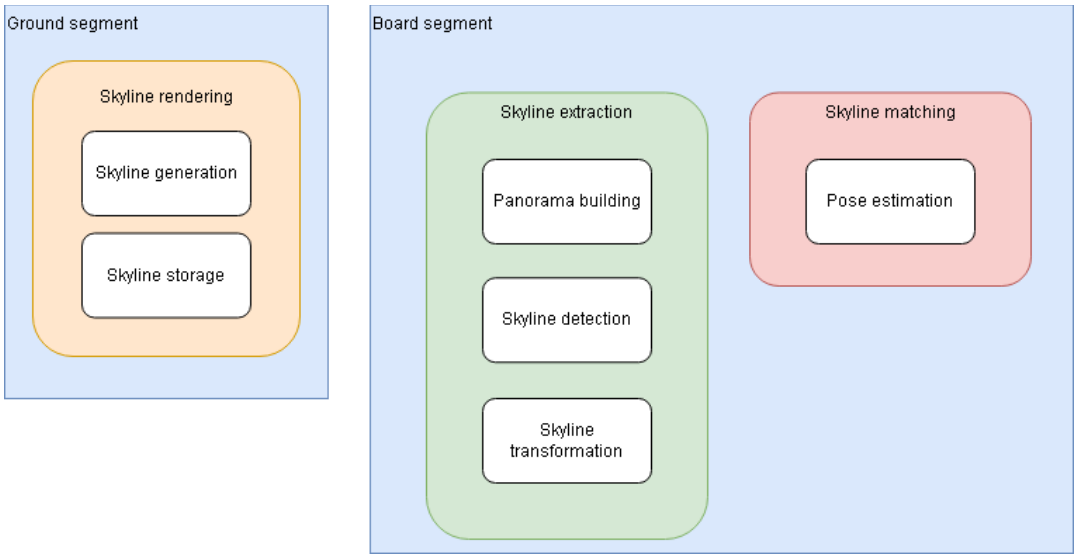


Figure 3: Subcomponent scheme for the mock-up version

9.1.1 Ground segment

The ground segment is constituted of one single component:

- **Skyline rendering:** the DEM is quantized according to a certain grid resolution and a skyline is extracted for every position of the grid and stored in a database. This step is done offline to allow for faster execution at run-time.
  - Inputs: DEM
  - Outputs: database of skylines for every grid position

The skyline rendering component is composed of several functions:

Subcomponent	Subcomponents	Algorithm description
Skyline rendering	Generation	Panoramic skylines are generated for every sample point in the DEM using image rendering and colour segmentation techniques in Blender.

	Storage	Skylines are encoded as vectors of elevations indexed by azimuth and stored in a database with their corresponding grid positions.

Table 2: Ground segment subcomponent descriptions

9.1.2 Board segment

The board segment is composed of 2 subcomponents:

- **Skyline extraction:** takes charge of assembling, extracting and encoding the skyline from a set of rover's images.
  - Inputs: set of rover's images
  - Outputs: panoramic skyline
- **Skyline matching:** the extracted query skyline is matched against all the precompiled DEM skylines and a position is estimated using a sum of squared errors metric.
  - Inputs: query skyline, database of rendered DEM skylines
  - Outputs: rover's position

Subcomponent	Subcomponents	Algorithm description
Skyline extraction	Panorama building	Fusion skylines taken at different orientations by stitching based on key-point detection, SIFT feature extraction, descriptor matching, homography matrix calculation and image spherical warping.
	Detection	Graph-searching via Dynamical programming using adjacencies, contrast cost and edge-ness.
	Transformation	The extracted skyline in image coordinates is converted to 2D elevation-azimuth coordinates and encoded into a vector of elevations indexed by azimuth.
Skyline matching	Pose estimation	Full grid search based on the sum of squared pixel differences for each possible azimuth

Table 3: Board segment subcomponent descriptions

9.2 SUBCOMPONENT DESCRIPTION

9.2.1 Subcomponent 1: Skyline generation

The skyline generation component consists in extracting synthetic skylines from the DEM according to a desired sampling resolution.

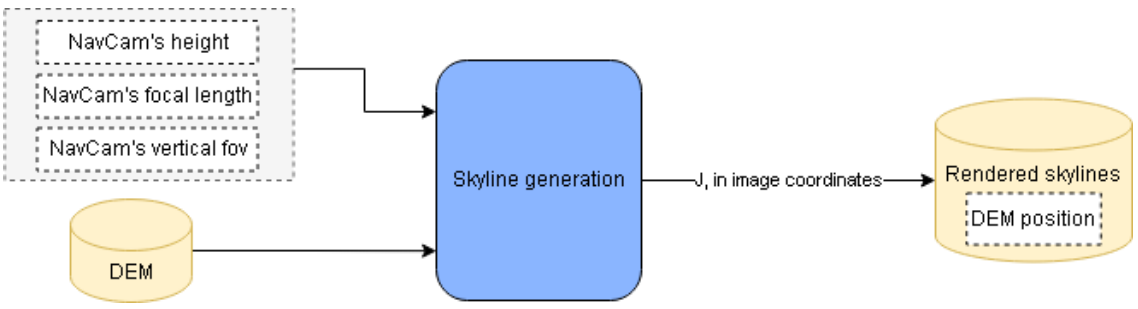


Figure 4: General architecture of the skyline generation component

It is composed of several functions:

- **DEM quantization:** this step consists in sampling the DEM with QGIS to obtain a 2D regular grid of possible candidate positions according to the desired resolution. This is done by using the GDAL translate command on the desired input GeoTIFF to extract the XYZ coordinates in ASCII format.
    - Inputs: DEM
    - Outputs: template positions grid
- **Panoramic image rendering:** black and white panoramas are rendered with Blender's equirectangular panoramic camera according to the NavCam's height and intrinsic parameters for every point of the sampling grid. A python script is used to automate the procedure of positioning the camera at the different points of the grid and rendering the panoramas.
    - Inputs: template positions grid, DEM, NavCam's height and focal length and vertical fov
    - Outputs: set of rendered black and white scenes in image coordinates
- **Colour segmentation:** colour segmentation techniques are applied to extract the skyline from the binary scenes.
    - Inputs: set of rendered black and white scenes
    - Outputs: set of rendered skylines in cylindrical coordinates

9.2.2 Subcomponent 2: Skyline storage

After generation, skylines need to be stored in a database along with their corresponding DEM position.

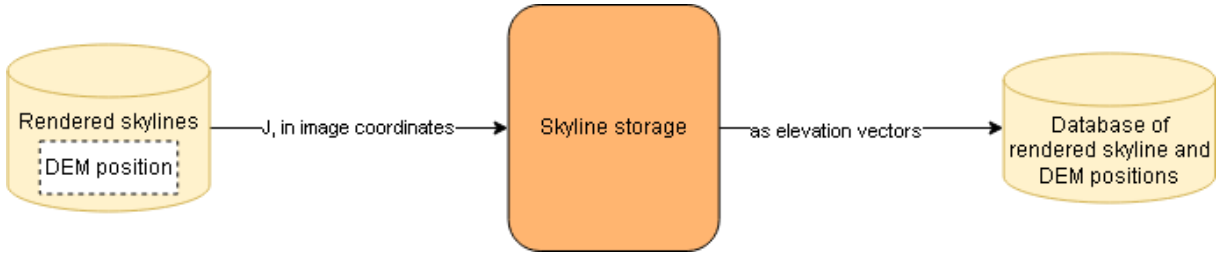


Figure 5: General architecture of the skyline storage component

The skyline storage component consists of several functions:

- **Coordinate transformation:** the generated skyline in image coordinates is converted to elevation-azimuth coordinates by using the vertical and horizontal panoramic FoV. For every column j, the elevation e_j and the azimuth a_j are calculated as follows:

$$e_j = \frac{e_0 - y_j}{h} \cdot vfov \qquad a_j = \frac{j}{w} \cdot hfov$$

Where e_0 is the panoramic image zero elevation line, y_j is the skyline row associated to column j, h is the height of the panoramic image, w is the width of the panoramic image, $vfov$ and $hfov$ are respectively the vertical and horizontal Fields of View associated to the panoramic image and expressed in [rad]. For the rendered skylines, $hfov$ corresponds to 360 and $vfov$ depends on the camera parameters. The resulting elevation vector ranges from $-vfov/2$ to $vfov/2$, while the azimuth goes from 0 to $hfov$. This azimuth is relative to the beginning of the image. However, the 0-yaw position of the panoramic image is stored and used for absolute azimuth calculation.

    - Inputs: set of generates skylines, vertical FoV, horizontal FoV
    - Outputs: set of rendered skylines in cylindrical coordinates

- **Skyline encoding:** from the set of skylines in cylindrical coordinates, vectors of 360 elevations are extracted by linear interpolation every 1°.
    - Inputs: set of rendered skylines in cylindrical coordinates
    - Outputs: set of rendered skylines as elevation vectors
- **Skyline storage:** the skylines are stored in a database as vectors of elevations indexed by azimuth, along with their corresponding DEM position.
    - Inputs: set of rendered skylines as elevation vectors, position
    - Outputs: database of skyline vectors with their DEM position

9.2.3 Subcomponent 3: Panorama building

A panorama is built from the set of overlapping skylines obtained through the previous step.

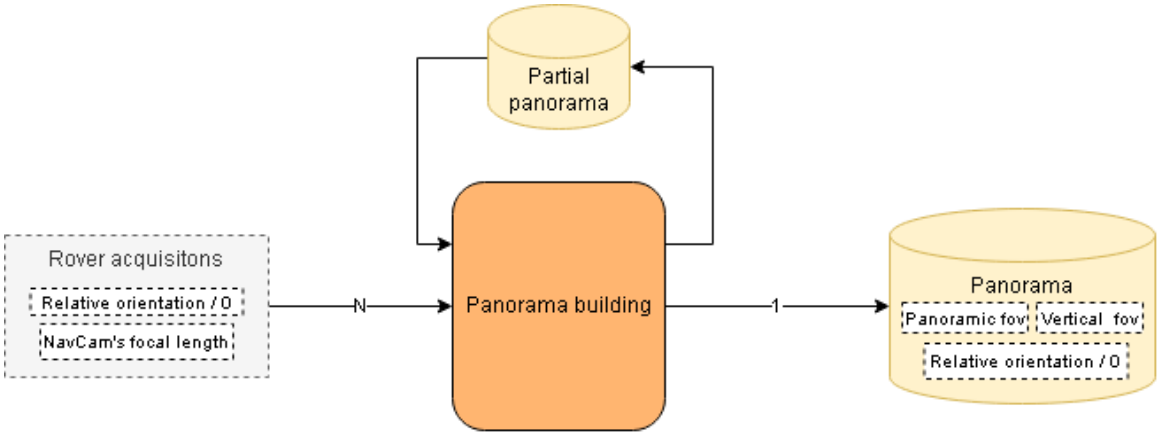


Figure 6 : General architecture of the panorama building component

- Inputs: set of rover acquisitions obtained for a particular rover position, NavCam's focal length
- Outputs: extracted panorama, panoramic image vertical and horizontal FoV, 0-yaw position

This step is performed using the OpenCV stitching function. It consists of a key-point detection step, followed by a SIFT feature extraction, descriptor matching between input images, homography matrix calculation, warping according to a spherical projection and final assembly. A preliminary function that did not deform the image but only transformed its coordinates was tested and found insufficient, as it did not account for any errors in the camera calibration.

As this new function transforms the images, black regions surrounding the panorama appear in the assembled image. A cropping step was performed to keep only the maximum inner rectangular region of the panorama. This step was inspired from [45] and it includes binary thresholding, contour extraction, bounding box computation of the largest contour and Region Of Interest (ROI) extraction. The horizontal and vertical Fields of View of the panoramic image are adjusted accordingly after image cropping and the 0-yaw position in the image is saved.

9.2.4 Subcomponent 4: Skyline detection

The skyline detection component consists in extracting the skyline from the rover's camera image. The extraction algorithm is based on graph-searching via Dynamical programming, using adjacencies, edgeness (whether a pixel is classified as edge or not by the detector) and contrast cost.

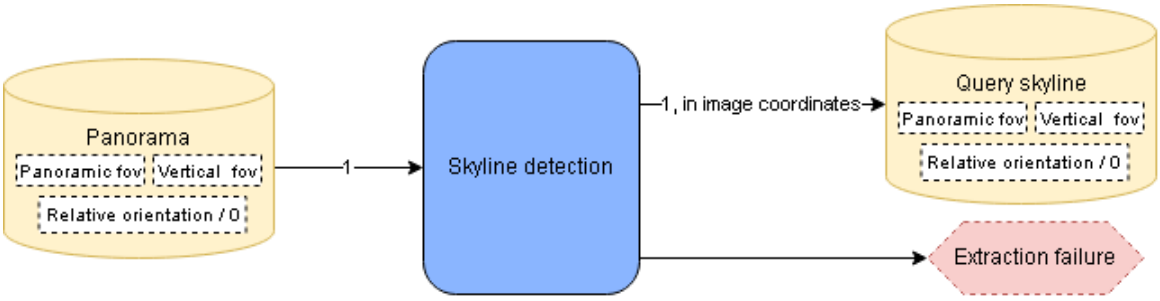


Figure 7: General architecture of the skyline detection component

It is composed of several functions:

- **Pre-processing:** the input image is converted to grayscale and Constrast Limited Adaptive Histogram Equalization (CLAHE) is applied. This method is use to improve contrast by equalizing the histogram in small regions of the image called tiles.
    - Inputs: extracted panorama
    - Outputs: pre-processed panorama
- **Edge detection**: the pre-processed input image is filtered with a Gaussian Kernel to remove the noise. Then, a Canny operator is applied to generate an edge map. The higher and lower thresholds of the edge detector are computed via Otsu binarization. Otsu's method determines an optimal global threshold value that separates pixels into 2 classes from the image histogram. This threshold is determined by minimizing intra-class intensity variance.
    - Inputs: pre-processed panorama
    - Outputs: binary edge map
- **Generation of gradient information**: a Sobel filter is applied to the pre-processed input image and the gradient's magnitude is obtained for each pixel.
    - Inputs: pre-processed panorama
    - Outputs: gradient information for all image pixels
- **Graph construction**: a dense multi-stage graph $G(V, E, \phi)$ is constructed, where V represents the set of vertices, E the set of edges and ϕ the cost function. It represents a weighted directed graph in which all pixels are considered as nodes and divided into different stages, corresponding to the columns. In this case, neighbours are considered at a maximum distance of 1 pixel either at the same or the next stage, so each node is connected to 5 other nodes in total.
  2 virtual nodes are added at the beginning and at end of the graph, connected to all nodes in the first and last column respectively. The cost function is defined as:

$$\phi(u,v) = \omega_1 * d\nabla(u,v) + \omega_2 * \big(1 - \nabla v(v)\big) + \omega_3 * B(v) + \omega_4 * (1 - h)$$

  Where $d\nabla(u,v)$ is the gradient difference between nodes u and v, ∇v is the absolute gradient value of node v, h is the node height and $B(v)$ is the value of the binary map at the pixel associated to node v:

$$B(v) = \begin{cases} 0 \; if \; v \; is \; not \; an \; edge \\ 1 \; if \; v \; is \; an \; edge \end{cases}$$

  All values are normalized. After some tweaking, the weights were set to 0.3 for the gradient difference, 0.05 for the absolute gradient, 0.3 for the edgne-ness and 0.35 for the height. The function maximizes the edge-ness of a pixel, minimizes the gradient difference between connected nodes, maximizes the height position in the image and maximizes in a small measure

the absolute gradient value. The gradient difference is prioritized compared to the absolute gradient value so as to avoid taking high-gradient pixels that are high in the image but do not belong to the skyline. Using the height inside the cost function could pose problems in common scenarios where clouds are present. However, this is not a problem in Mars.

  - Inputs: Binary edge map, gradient information for all image pixels
  - Outputs: Multi-stage graph (vertex, edge points, cost function)
- **Graph resolution** via a shortest path algorithm (Dijkstra) between the starting (source) and end (sink) virtual nodes.
  - Inputs: multi-stage graph
  - Outputs: query skyline in image coordinates

9.2.5 Subcomponent 5: Skyline transformation

The extracted skyline in image coordinates, expressed in pixels, is converted to a 2D cylindrical coordinate system. Elevation is expressed as a function of the azimuth.

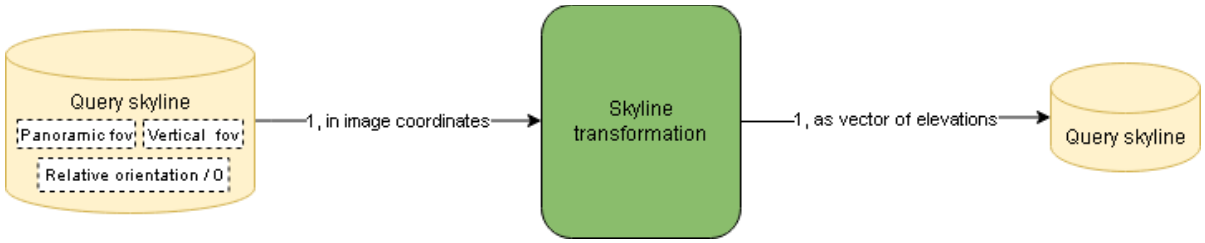


Figure 8: General architecture of the skyline transformation component

The skyline transformation component consists of 2 functions:

- **Coordinate transformation:** the extracted skyline in image coordinates is converted to elevation-azimuth coordinates by using the vertical and horizontal Fields of View of the panoramic skyline. The formulas are identical to the ones used for skyline generation from the DEM. However, in this case, the Fields of View need to be adjusted according to the panorama coverage and its posterior cropping.
  - Inputs: query skyline in image coordinates, panoramic image vertical and horizontal FoV
  - Outputs: query skyline in cylindrical coordinates
- **Skyline encoding:** the skyline expressed as elevation-azimuth is sampled every 1° by linear interpolation to obtain a vector of elevations indexed by azimuth.
  - Inputs: query skyline in cylindrical coordinates, adjusted
  - Outputs: vector of elevations for the query skyline

9.2.6 Subcomponent 6: Pose estimation

The pose estimation subcomponent consists in using a **pyramidal approach** over the space of position candidates and evaluate the match between the corresponding skyline from the DEM and the extracted query skyline. As images have been previously corrected using the proper camera's calibration file, a Bayesian approach which considers a Gaussian model for the measurements and uses a standard deviation as an indicator of sensor quality does not seem appropriate. Thus, a simple measure which considers the sum of squared errors for each column is judged sufficient. For each extracted rover's skyline, the pixels are compared to the generated DEM skylines. All azimuths are evaluated. To make the

algorithm robust to errors in the camera height, the elevation difference with respect to the mean (for the rover's panoramic image horizontal FoV) is compared instead of the absolute elevation value. This approach allows to compare partial panoramas, as the mean is computed every time for the corresponding portion and the tested azimuth.

For the azimuth estimation, the position of the center of the image taken at 0 yaw angle is considered.

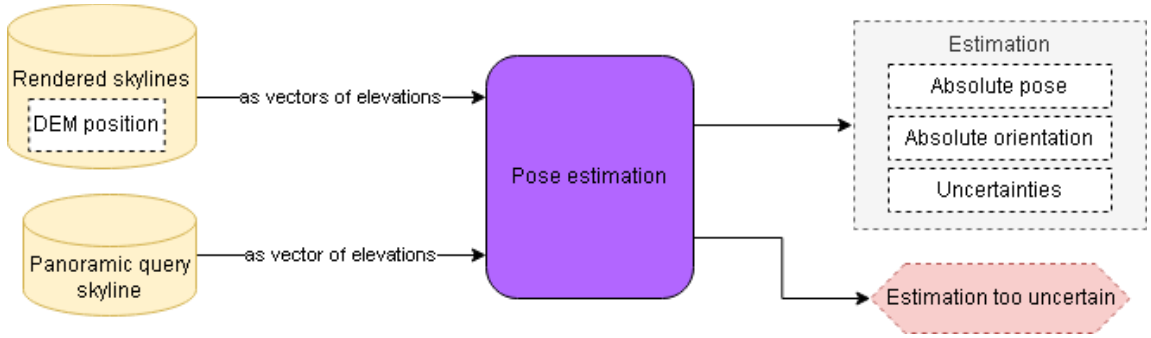


Figure 9: General architecture of the position estimation component

- **Skyline matching:** the matching score is computed for a candidate grid position to be the real rover's position. The matching score corresponds to the sum of squared elevation errors between the measured skyline and the rendered skyline, considering the azimuth estimation that minimizes the error. In consequence, all possible azimuth values are tested. If p is the tested grid position, e_j the elevation value corresponding to azimuth j for the extracted skyline, s_j is the elevation value corresponding to azimuth j for the rendered skyline, and $\alpha^*(p)$ is the absolute azimuth estimation that minimizes the score for the position p, the error score can be expressed as follows:

$$\epsilon^2(p,\alpha^*(p)) = \sum_j (e_j - s_j(p,\alpha^*(p)))^2$$

  Where $\alpha^*(p) = argmin_\alpha\, \epsilon^2(p,\alpha)$.
  - Inputs: rendered skyline, extracted query skyline (both expressed as vectors of elevations)
  - Outputs: matching score, optimal azimuth estimation
- **Pose estimation:** a pyramidal approach is applied to estimate the position and compute a measure of uncertainty from a set of matching scores and estimated azimuths. A first estimate is obtained with a bigger search area and step size, which is then refined using a smaller search area and smaller step size in a 3-step process. The best candidate location after each step is the one that has the lowest error:

$$(p^*,\alpha^*(p^*)) = argmin_{(p,\alpha^*(p))}\epsilon^2(p,\alpha^*(p))$$

  The final estimate is the best candidate after the last step.

  - Inputs: set of matching scores and azimuths
  - Outputs: rover's position, measure of estimation uncertainty

10 IMPLEMENTATION PROCESS

10.1 SKYLINE RENDERING FROM A DEM

The first part of the skyline matching algorithm is to render the skylines from a DEM. To start this process, an appropriate DEM was needed. A Digital Elevation Model is a representation of the terrain's elevation at every point. It can be seen as a function $z(p)$ that provides an altitude value for every grid position $p = (x, y)$ according to a certain resolution. For the sake of efficiency, the DEM had to serve to test the algorithm on both real and simulated data, it had to be precise enough to allow 1º skyline resolution, and big enough to allow testing the algorithm for the different scenarios. A DEM of Bardenas, Spain, was acquired through the Spanish National Centre for Geographic Information (CNIG) website[6]. DEMs of all the territory are available in 2, 5, 25 or 200m resolution. They are subdivided in many different tiles and can be obtained in ASCII format, either by selecting a point or polygon or by searching a coordinate, a parcel, an administrative division or the tile number.

A rectangular area of 110x95km with 2m resolution was selected around a chosen site in Bardenas. The size of the area was roughly estimated by using the 3D view in Google Maps and Google Earth to get the location of the farthest horizon from certain points and then trace the distance with the 2D view. A compromise had to be made in terms of horizon inclusion since, from a certain distance, the horizon is not clearly captured using real cameras and thus a skyline extraction algorithm cannot discern it either.

The ASCII tiles were fused together and transformed to GeoTIFF rasters using QGIS. QGIS is an open-source Gegographic Information System (GIS) software that allows managing geospatial data via the GDAL library. GDAL is a library used for reading and writing geospatial rasters and vectors. The GeoTIFF was loaded in Blender, with the BlenderGIS plugin. Blender is a modelling, animation and 3D rendering software with a powerful render engine and many different features like UV mapping, texturing, particle simulation, fluid and smoke simulation, sculpting or compositing. The BlenderGIS plugin helps import and manage georeferenced data.

At this point, many problems arose as the computer could not load the DEM due to being high-resolution and very large. Thus, the resolution was decreased to 5m. Nevertheless, problems kept arising either when loading or when trying to render images from the DEM. This led to think that the 5m resolution zone had to be kept to a minimum, so a multi-resolution DEM was generated, adapting the idea of [12].

Several distances were defined from a chosen search area and their minimum resolution to obtain 1° azimuth accuracy was calculated. 4 different DEMs (3x3km with 5m resolution, 5x5km with 26m resolution, 14x14km with 44m resolution and the rest with 122m resolution) were finally generated via QGIS and loaded into Blender. A python script was coded to subtract the overlapping areas from the lower resolution DEMs, using GDAL's translate, vrt transformation, calculation and warping functions.

[6] https://centrodedescargas.cnig.es/CentroDescargas/busquedaSerie.do?codSerie=MDT05

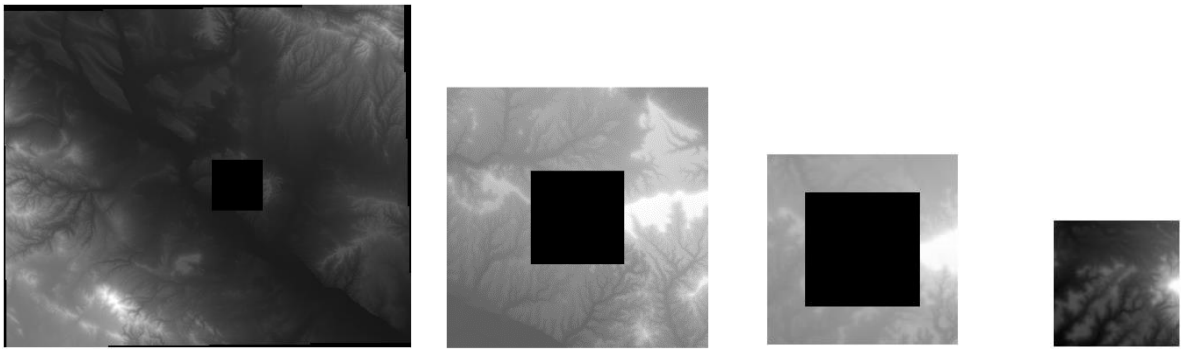


Figure 10: Generated rasters using QGIS (not scaled)

A 7x7 km grid with 5 m resolution was generated using QGIS and saved in a file to be used as search area. The 4 DEMs were loaded in Blender, their material was modified so as to emit white light and the background colour was set to black. This allowed to render binary images. The panoramic equirectangular camera was added and its parameters were set according to the Magellium's robot camera.

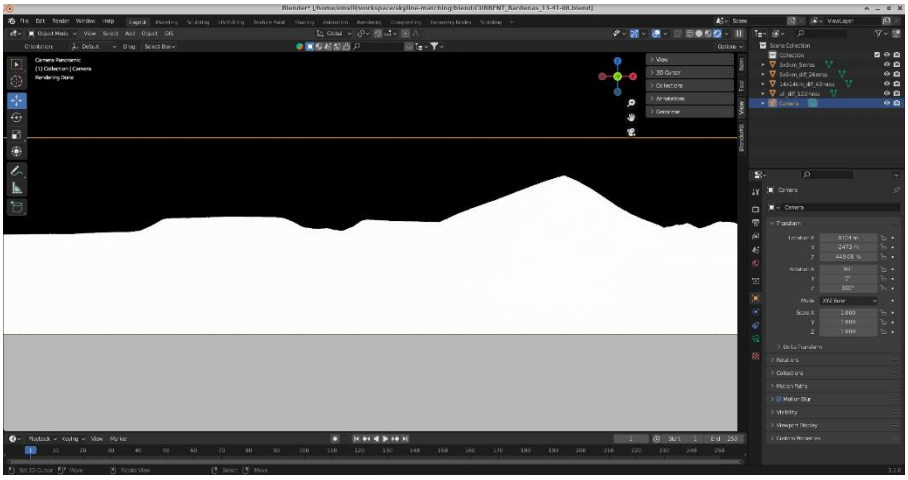
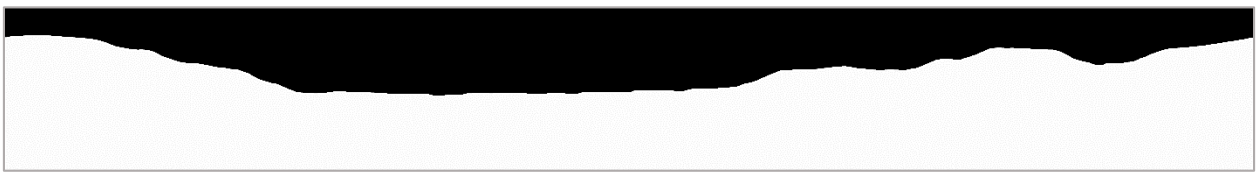


Figure 11: Blender interface and camera view

A python script was coded inside Blender to automate the process of setting the good camera position and orientation, rendering the panorama, extracting the skyline and saving the vector for every grid position. When starting the automated rendering process, the estimated rendering time was over 40 days. Due to time limitations, the search area had to be reduced to 1x1km and modifications had to be made to improve the rendering time, like reducing the samples, removing unnecessary options like denoise, reflection or refraction, reducing the number of light bounces, etc.

In the end, the skylines could be rendered in about a week. However, as said, the testing area for the after-landing scenario had to be changed to 1x1km and the maximum DEM resolution to 5 m instead of 1 as previously specified. A rendered panorama and its extracted skyline are shown below.



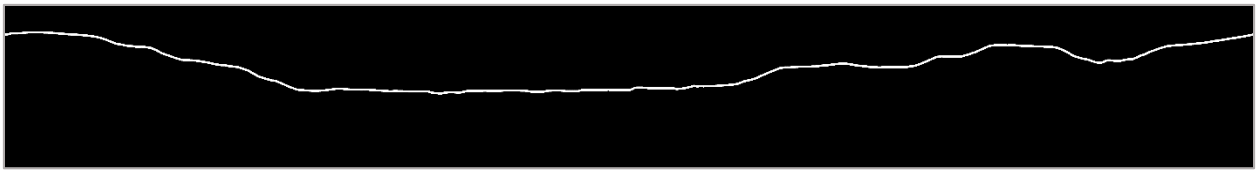


Figure 12: Rendered panorama and extracted skyline

10.2 CODING THE ALGORITHM

To code the algorithm in C++, I had to learn Magellium's practices and guidelines in terms of project architecture, coding rules, dependencies, COTS, CI/CD, GIT and docker. I had to learn how to use the internal libraries for thread management, log creation and configuration file management as well as the creation of executables. It should be noted that I had very little experience in compiled programming languages so this was an added challenge. Luckily, my team mates help me a lot during this phase that started with quite a few compilation errors. Next, the procedure for the different parts of the algorithm is explained.

10.2.1 Panorama assembly

In terms of panorama assembly, a simple version was first implemented, with coordinate transformation and blending based on overlapping FoV. This was thought to be enough as the images were rectified after camera calibration. However, this was shown to be insufficient due to potential errors in camera parameters. Thus, a new version using openCV's stitching function for panoramas was used, with key-point detection, SURF feature extraction, feature comparison between input images, spherical warping and blending. As observed, this leads to an equirectangular panorama, like the one rendered in Blender.

Figure 13: Panorama assembled

At first, some problems appeared with this new approach as it seemed to work only with certain panoramas or, in some cases, it missed some images. After extensive testing, this was solved by reducing the confidence threshold for two images to be from the same panorama.

In the final version, when testing with simulated data covering a 360° panorama with high overlap between images, some cases were reported where the panoramas did not get assembled in the right orientation, meaning that the images were not assembled in order. Even though the panoramas were still well assembled and the good position was found, the orientation estimate was not correct. These cases were very particular and were eliminated from the tests. However, correcting this is one of the main priorities for future improvements. The most probable solution would be to perform the different stitching steps separately, image by image, instead of using the stitching openCV function for panoramas.

10.2.2 Skyline detection

The skyline detection step was by far the hardest part of the C++ algorithm, as many different conditions needed to be considered to ensure a good extraction (incomplete skyline, blurred skyline, occluded skyline, high gradients close to the skyline like clouds or vegetation…). An approach robust to all these possible perturbations was sought, and this is why a dense graph was adopted. This approach was inspired from [22]. Since this method evaluates all pixels, and not just those that are edges, it can find the good skyline even if it is incomplete or occluded in some parts (or if it does not start in the first column). Moreover, seed selection and gap filling are not required, in opposition to edge-based methods.

Originally, the implemented resolution method consisted in finding all available paths from the nodes of the first column to those of the last and to keep the minimum cost path. However, this approach was very expensive computationally, so the image had to be degraded before the detection to allow faster speed. Nevertheless, the results were not optimal. In the end, virtual nodes were added at the beginning and the end of the graph, which acted as source and sink nodes for the Dijkstra shortest-path algorithm. This drastically speeded up the algorithm and so the image could be kept at full resolution during all the procedure.

At first, connections with neighbouring nodes were only allowed from a stage to the next. With this approach, high slopes could not be detected, due to several skyline pixels being located in the same column. Thus, connections were permitted also with nodes at the same stage.

At this point, although the algorithm was able to find the skyline in most cases, there were some occasions in which some peaks were missed due to a very high gradient being too close to the peak. Tweaking the graph parameters did not help much and, thus, a pre-processing step was added. Different things were tried, like enhancing the contrast and applying morphological operations, but they did not help much. Finally, CLAHE was applied and it made a big difference in these situations. The final algorithm is represented in Figure 14.

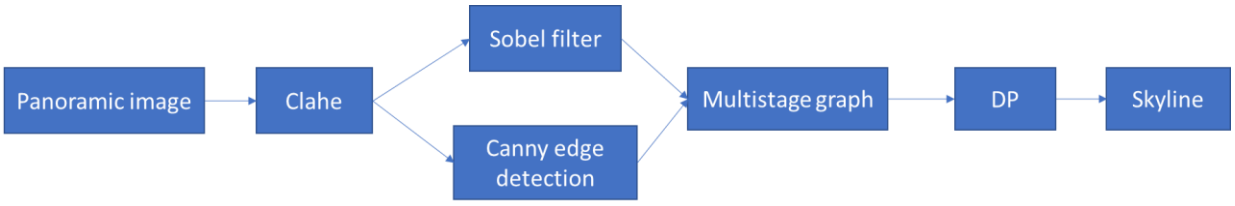


Figure 14: Steps of the skyline detection algorithm

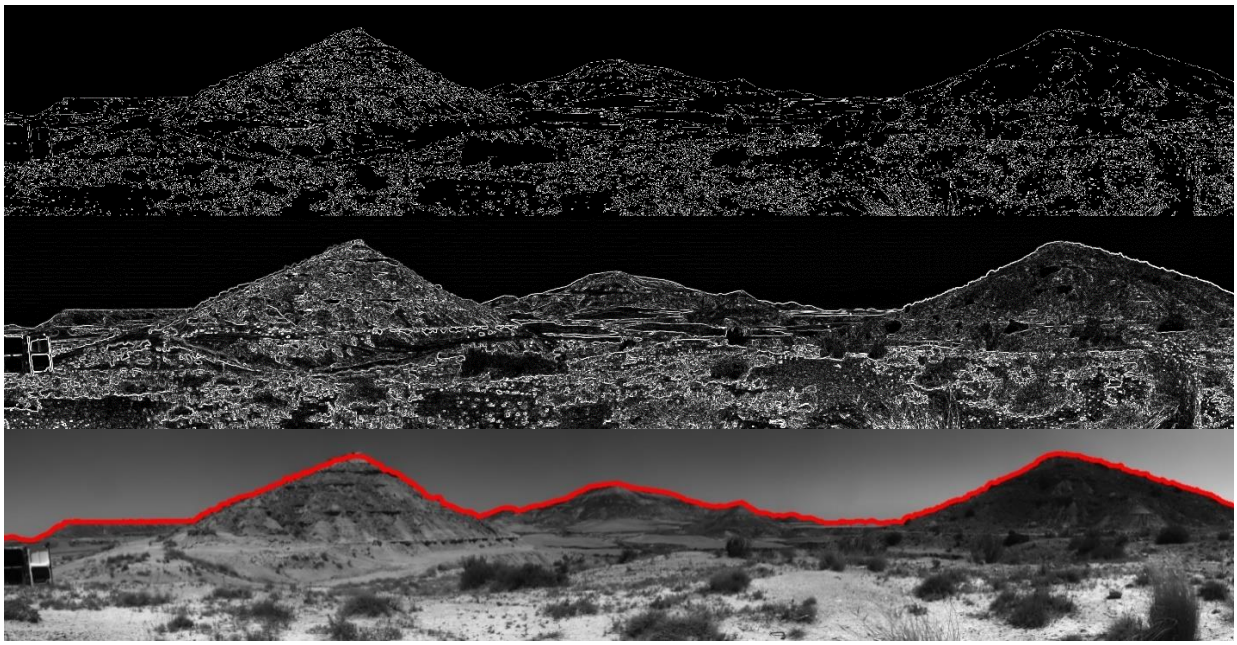


Figure 15: Image after clahe (row 1), Canny edge detection (row 2), gradient extraction (row 3) and skyline extraction (row 4)

Even though the final skyline extraction is quite satisfactory, the highest part of one of the peaks is not accurately detected, as can be seen below. After intensive testing and tweaking, this was the best result that could be obtained. As stated before, the high gradient present at the upper part of the peak, created by its natural form and texture, makes it hard for the algorithm to work properly. Additionally, in this case, the gradient between the mountain and the sky is not high around this area.

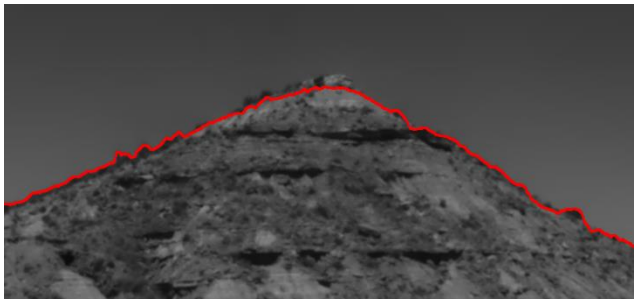


Figure 16: Skyline detection at the peak

10.2.3 Skyline transformation

The skyline transformation component was the simplest to code, since it is based on a simple coordinate transformation followed by a linear interpolation. The resulting vector can be found below, next to the complete ground truth vector in the same axis scale for the closest grid position (less than 1m away). The extracted vector has been represented according to its ground-truth orientation for comparison purposes. As can be seen, the shape is quite similar when considering the right portion of the plot. Nevertheless, it can be seen that the main peaks have a lower value for the extracted skyline plot, probably because of the detection imprecision stated before. However, the main difference lies in the area between the 2 main peaks, which are much lower and more irregular in the ground-truth plot.

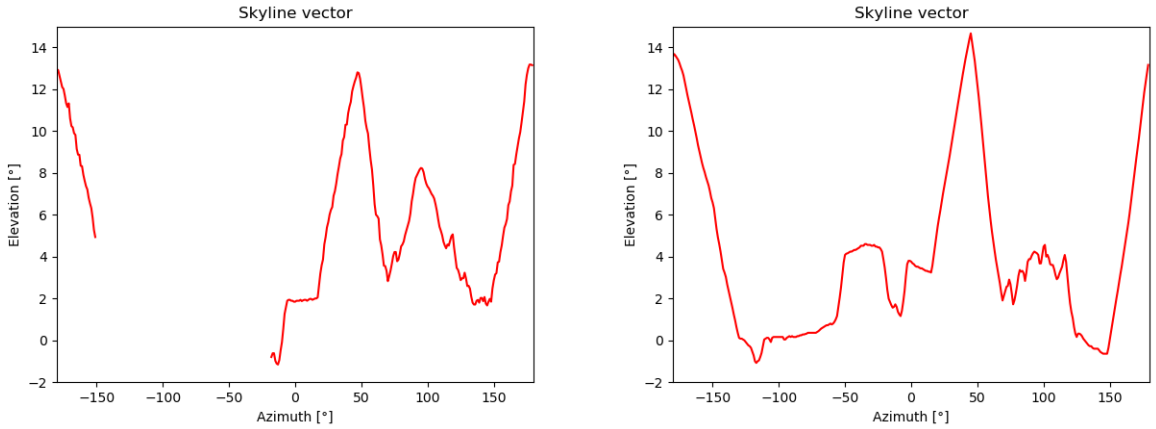


Figure 17: Encoded skyline vector (left) and ground truth full skyline vector (right)

10.2.4 Pose estimation

For the pose estimation component, a Bayesian approach was originally implemented. However, since the images were rectified, implementing a measure of sensor quality and assuming Gaussian measurements did not seem appropriate. Therefore, a simple sum of squared errors was adopted instead.

The main problem regarding this component was related to the pyramidal approach. Basically, the score did not evolve as expected, making the pyramidal approach highly dependent on the starting position and other "random" parameters. This makes it inadequate for evaluating skylines that are relatively close to the camera. For that reason, it was not used in the end. For more details refer to the test in section 12.1.1.

A parameter was later added to the pose estimation function to allow testing the 3rd scenario. If this parameter is set to true, the pose estimation function only computes the orientation for a certain input position.

The score contour plot for the previous skyline is shown in Figure 18. For reminder, the tests have been performed on real data, with the skyline extracted in the previous sections. The error in position is 4.52m for x and 0.55m for y. The error is computed with regards to an imperfect GPS data, since the RTK base did not converge during the field tests. The values are expressed in the UTM ETRS89 projection zone 30N.

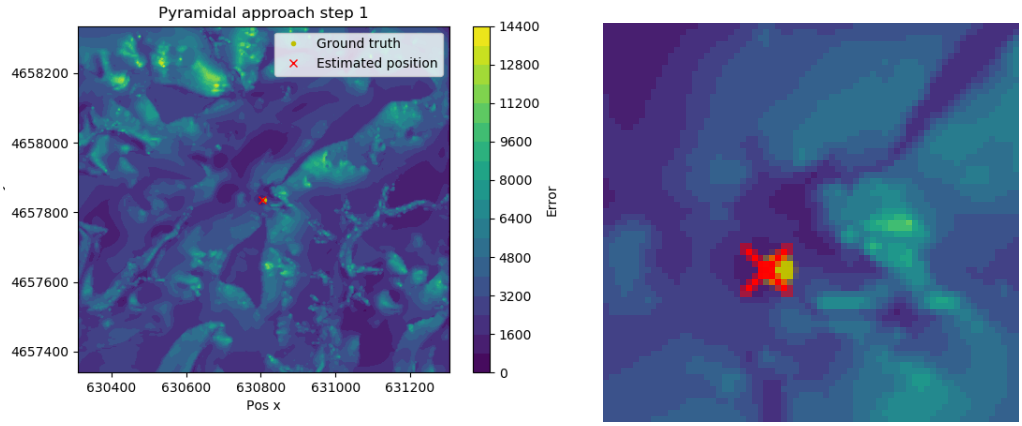


Figure 18: Contour plot of the score function (left) and zoom on the ground-truth position area

For comparison purposes, the skylines for the closest and the estimated position are shown below. It can be noted that, by sight, it is almost impossible to distinguish them. However, by looking at the values one can see that the main peak values are lower for the estimated skyline while the middle region values are higher.

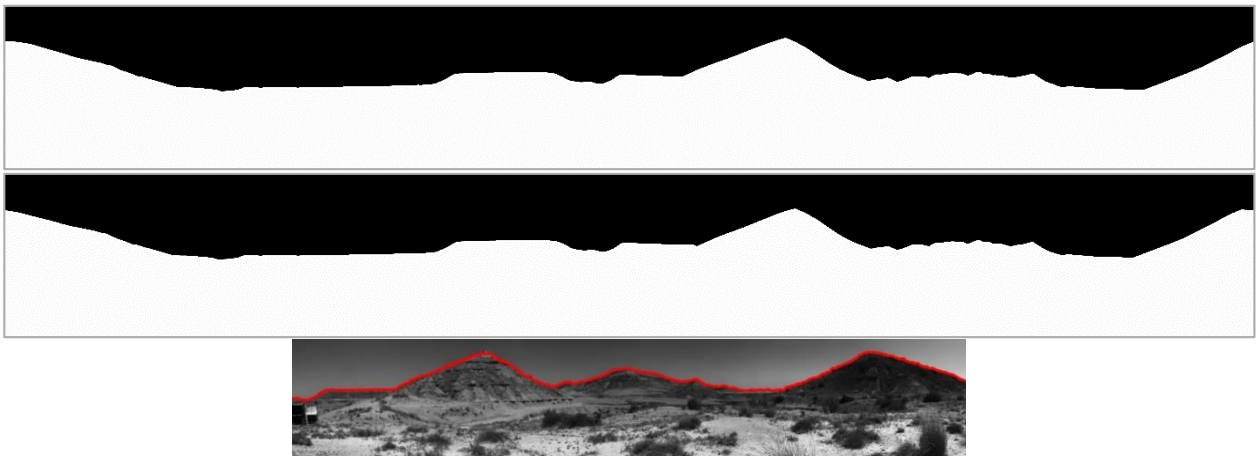


Figure 19: DEM ground-truth skyline (1st row), estimated DEM skyline (2nd row) and extracted skyline (3rd row)

Below, the extracted skyline is overlapped to the ground truth skyline for the estimated orientation. The most remarkable thing is that the extracted skyline is higher than the other two on all the peak areas, and the middle peak has a greater slope. This is likely due to a slight pitch and roll variations in the acquisitions with respect to the rendered skylines.

Figure 20. Extracted skyline overlapped to the ground-truth rendered panorama

11 DATA ACQUISITION

11.1 BARDENAS FIELD TESTS

I was involved in a 3-day scouting trip to Bardenas, in Spain, with the project leader and another member of the IA unit who was responsible of operating Magellium's acquisition platform. The objective of this trip was to test the robot on representative terrain, scout sites for a future testing trip and get acquisitions to test my algorithm on real data. The trip took place between July 4th and July 6th.

The main requisite for the chosen site was that it had to be representative of real Mars fields. Its global required characteristics were the following:

- Deserted land, with no prominent vegetation or artificial constructions.
- Mountainous irregular terrain (holes, mounds, flat areas, steep cliffs…).
- Ground made mainly of sand and rocks.
- Dusty environment.

In terms of size, the site had to be big enough to allow for a 7x7 km search area to test the after-landing scenario. The main feature of interest was the relief, so different types of terrain were sought (flat, hills, cliffs, holes…). The objective was to have different horizon types and to be able to set the rover on different terrains.

The main constraint for the site was that it had to be easily accessible from Magellium's headquarters, to allow for easy equipment transportation and low-cost travel. Additionally, a precise enough DEM for the site had to be available due to limited time, resources and authorizations.

11.1.1 Material

The material used for the field tests was the following:

- Magellium's rover
    - Stereo-bench type ExoMars/ Mars 2020 NavCam
    - Steerable PTU with orientation values (1° accuracy)
    - Sensors (IMU, wheel odometers) type ExoMars/ Mars 2020 rover
    - Data storing capacity
    - 2 antennas GNSS-RTK
    - Module GNSS
- Base GNSS

Magellium's robot is shown in . Its stereo-bench specifications, compared to ExoMars and Mars 2000 NavCams, can be found in Annex 4: Magellium's test rover stereo-bench specifications

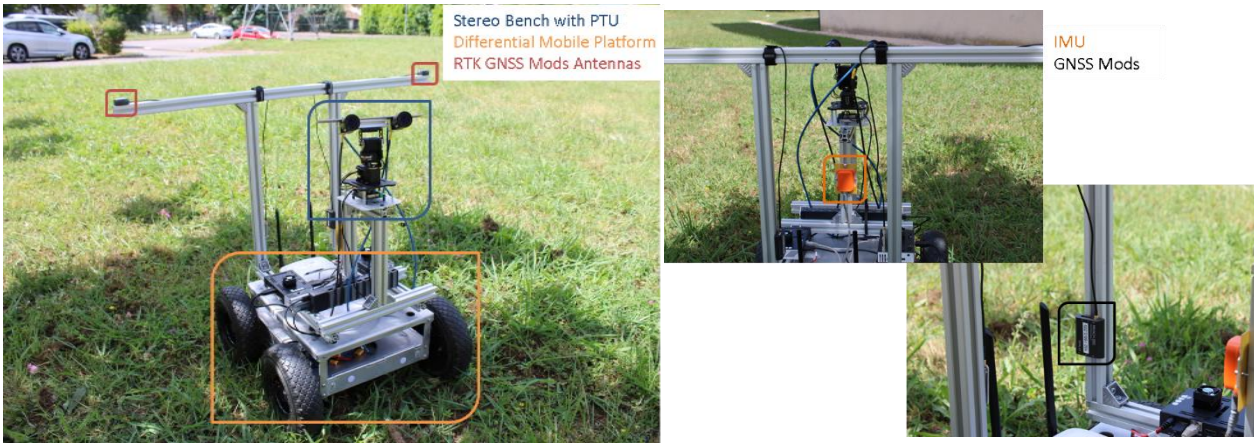


Figure 21: Magellium's test rover

11.1.2 Acquisitions

For each testing site, the robot, the GPS-RTK base and the computer had to be set up and connected. First, a set of acquisitions was generated with a checkerboard covering different positions in the image for posterior camera calibration. Then, a panorama was launched, in which the camera rotated and took pictures according to the pre-set angles. 5 acquisitions were generated for each site, covering about 242°. Full 360º panoramas could not be covered since the 2 GPS-RTK antennas would have been in the camera's field of view for larger panoramas.

Figure 22. Magellium test rover in Bardenas

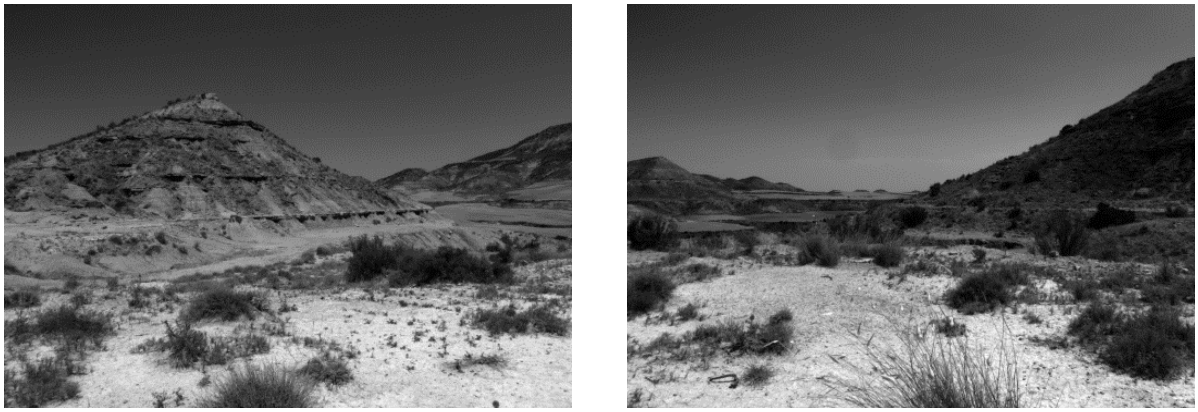


Figure 23: Examples of acquisitions obtained in Bardenas

Several problems were encountered on-site. Mainly, one of the GPS was not working. As heading computation was meant to be performed on the basis of 2 GPS antennas, orientation measures could not be obtained. Additionally, with this method, pitch and roll could not be computed, which implies added difficulties for the algorithm as images cannot be rectified accordingly. Finally, the IMU was not calibrated, so a measure of the angle was not available. Another issue was that the GPS-RTK base did not converge so the accuracy of the obtained position cannot be guaranteed.

After the data acquisition phase, the files had to be converted in order to extract the metadata (intrinsic camera matrix, relative camera orientation, panorama id, frame, image characteristics…) and the ground-truth GPS data in the right format.

11.2 SIMULATED DATA GENERATION

Due to the aforementioned issues, generating simulated data was a crucial step to allow testing the algorithm with different conditions and in different places.

This step of the algorithm was performed using Blender, with the same multiresolution DEM of Bardenas described before. The camera was configured according to the real robot camera's characteristics and height. The DEM material was changed to make it more realistic and several features, such as rocks, pebbles, stones or small plants were added so as to allow the panorama assembly step to work. The sky was configured with a particular texture, and several parameters regarding the air, dust and ozone density were changed, as well as the sun position and elevation. Different acquisitions were been performed by rotating the camera over the same position within a Python script. 13 acquisitions with about 50% overlap were generated for each position.

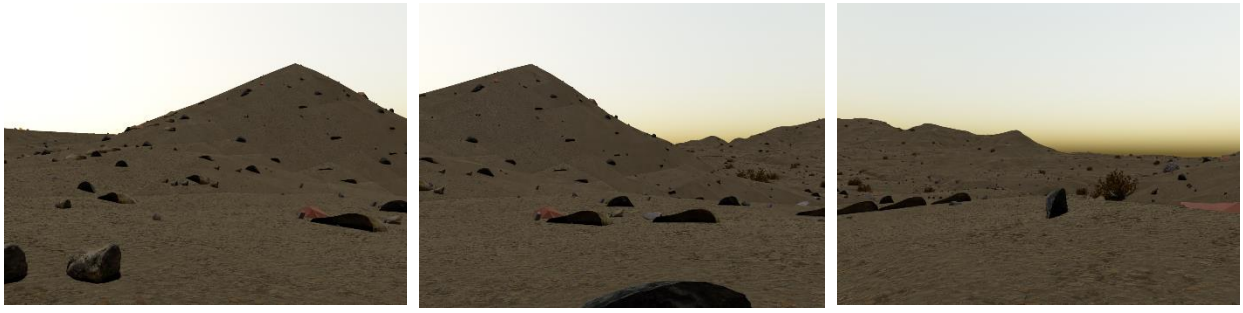


Figure 24: Generated simulated acquisitions

Even with the addition of features and the increase in overlap, the panorama assembly component appeared to have a hard time for most sets of simulated acquisitions. Thus, simulated panoramas were generated with Blender's equirectangular panoramic camera. The aim was to test the detection, transformation and pose estimation components. Several tests were executed using different levels of noise (perfect, slightly, or really blurred). The perfect data corresponds to the binary panorama, the slightly noised corresponds to the addition of particles like pebbles or rocks and even some vegetation. Finally, the highest level of noise corresponds to a lens distortion and dispersion added to the image through the addition of a compositing node.

12 RESULTS

Several tests were performed to evaluate the quality and the performance of the algorithm and its different components. Due to limited time and resources, real acquisitions were used for only one testing location. The rest of the tests were performed using simulated data.

In all the tests, the DEM used is the multiresolution DEM described in section 10.1. The resolution of the DEM inside the search area is 5m. The grid size and the grid resolution are specified for each scenario.

Different tests were performed in order to evaluate all testing scenarios.. As a reminder, the projection used to evaluate the error in position is the UTM ETRS89 zone 30N (EPSG:25830).

12.1 SCENARIO 1: AFTER LANDING

The lost-in-space type scenario was evaluated on a search area of 1x1 km with 5m DEM and grid resolution. As mentioned before, the size and resolution of the search area were reduced due to computational constraints. A real acquisition was used to test the performance of the whole algorithm and several tests were performed using simulated panoramic images rendered via Blender. Different level of noises and other parameter variations were introduced to test the robustness of the extraction, transformation and pose estimation components.

12.1.1 Test 1: Pyramidal approach, grid size determination

Unitary tests were performed on the pose estimation component using ideal rover vectors extracted from the DEM and a 1x1km search area with 5m resolution. This was done to observe the score evolution and determine the appropriate grid size for each step of the pyramidal approach.

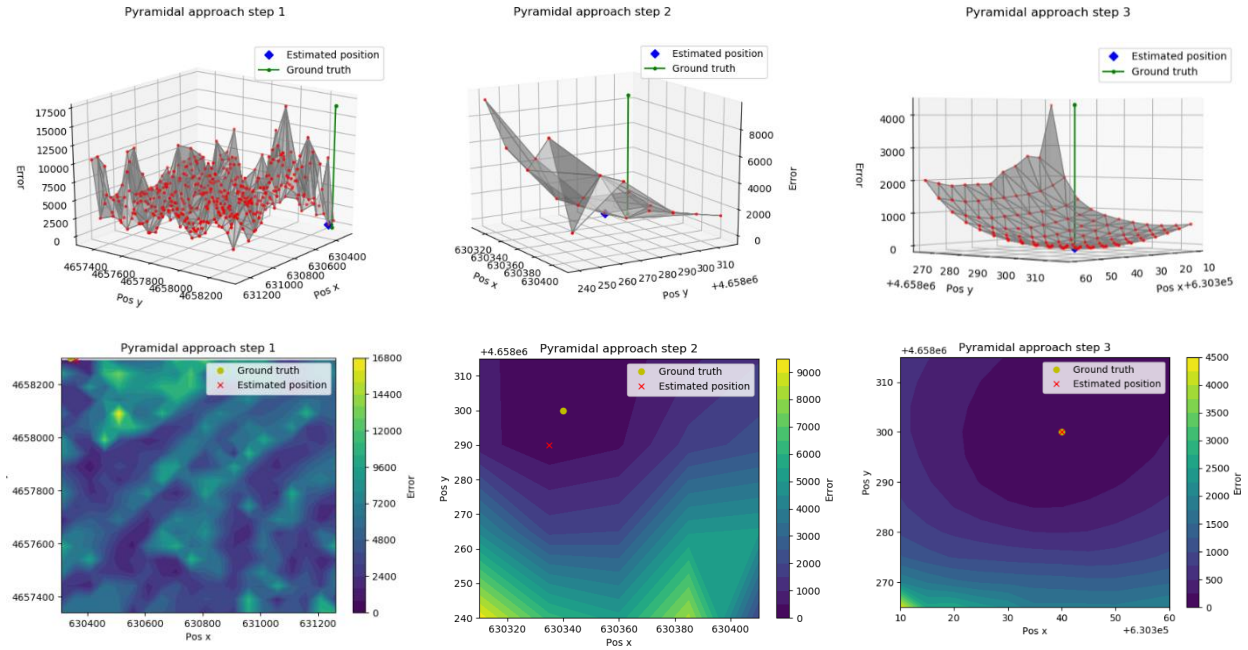


Figure 25: Pyramidal approach with 50, 25 and 5m step sizes (from left to right)

As observed, the scores don't follow a global pattern and they exhibit a chaotic behaviour on a large-scale. However, close to the appropriate skyline a descending parabolic-type of shape can be observed. This leads to think that a certain step size can be chosen to make sure that at least one testing position will fall inside the area where the score is lower than at any other local minimum. The objective was to determine the size of this area.

After testing with several skylines, a problem was identified: although this reasoning seems to be valid for relatively far skylines, skylines that are fairly close the camera change significantly within a few meters of distance. For the tested skyline in question, the good position was only found with an initial step size of 10m. The graphics show that the scores vary notoriously with only 5m of distance.

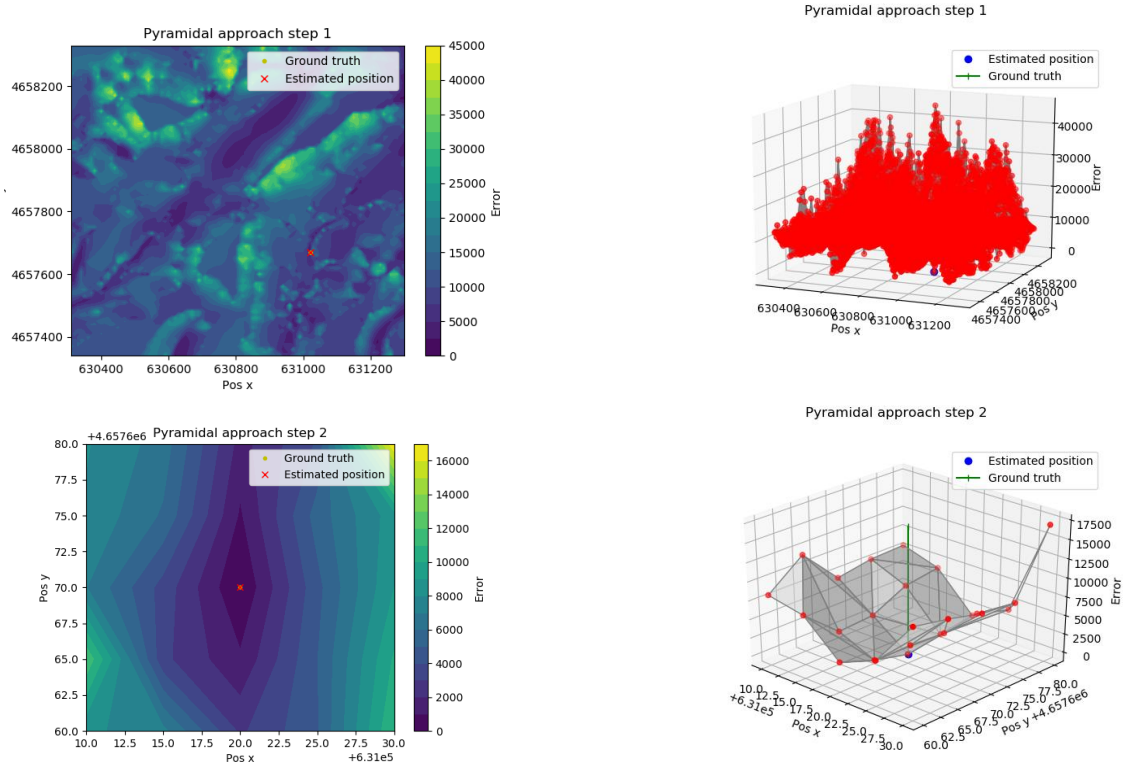


Figure 26: Pyramidal approach with 10 and 5m step sizes

The same problem was found when testing the whole algorithm with a real acquisition. Only when setting an initial 5m grid step, the position with the minimum error was found.

Overall, even though the pyramidal approach is highly appropriate for evaluating certain functions, it might not be suitable for evaluating skylines, where the score is highly unpredictable and presents many local minimums. Nevertheless, this approach can be very promising for speeding up the computational time if modified accordingly. A modified pyramidal approach, where mean skylines are rendered from several different zones of the grid, is contemplated for future improvement. The idea is that, for each pyramidal level, a "mean" skyline vector represents a certain area of the grid and takes into account all of the information of its subgrids present at the next pyramidal level. Another option would be to consider several candidate positions after each step.

Since the pyramidal approach, as implemented currently, can lead to false global minimums, it is not going to be used to test the performance of the algorithm in the rest of the tests.

12.1.2 Test 2: Robustness to different horizon types and noise levels

The goal of this test was to test the robustness of the extraction, transformation and pose estimation components to different levels of noise and different horizon types (flat and distant, close and prominent or highly irregular relief). To achieve this, 3 candidate positions and 3 levels of noises were chosen to generate the simulated data. All candidate positions were located in between grid positions, as would happen realistically.

The different levels of noise correspond to:

- 1: Perfect binary panorama
- 2: Realistic panorama, with rocky sand DEM texture, dusty sky texture, and added pebbles and stones that appear over the horizon line.
- 3: Realistic panorama with added lens distortion and dispersion, simulating real camera noise or other deformations that could have potentially been introduced by the panorama assembly part.

The different panorama ids correspond to:

- 1: flat and distant skyline
- 2: Prominent close skyline
- 3: Irregular fairly distant cliffs

The panoramas used and the superimposed extracted skyline can be found in annex 1.

Id	Noise level	Error pose x (m)	Error pose y (m)	Error heading (°)	Error score
1	1	1.5	3	0	1.1
	2	1.5	2	0	9.61
	3	1.5	13	12	50.04
2	1	0.75	2	0	206.65
	2	0.75	2	0	191.51
	3	19.2	28	8	339.95
3	1	1.5	2	0	25.93
	2	1.5	2	0	27.63
	3	3.5	13	0	534.31

Table 4: Results of the robustness to horizon types and noise levels

The fact that even the binary panoramas, considered as "ideal", have some error in position is a result of the data being rendered at positions that are not part of the quantized grid. As the grid resolution is 5m and the error is less than 2.5m in all cases for the first and second levels of noise, this is the minimum error that could have possibly been obtained.

For the second set of panoramas (id 2), the second level of noise presents a lower error than the first, even though some rocks have been taken as part of the horizon. This is potentially due to the fact that, at the closest grid position, the mountains are a bit closer to the camera than at the actual camera position and, thus, they appear a bit higher in the image, even though the difference is very small.

In terms of localisation accuracy, the algorithm achieves its best possible results with the first 2 levels of noise. However, the results show that noise, particularly camera noise or reflections, largely affect the performance of the algorithm. In terms of extraction, the algorithm tends to be quite robust to elements located over the horizon, like rocks.

For the strongest noise level, in terms of horizon type, the best results are unexpectedly obtained with the flat and distant type. One could think that, as the skyline is not very distinct, the algorithm could have found many different positions that are visually similar around the same area. The second best result is obtained with the irregular fairly distant hills. This would have been intuitively the best-case scenario, since the skyline is distinct but it does not change a lot over a very small camera displacements, allowing for some errors. The results for these two panorama types are quite close. Clearly, the worst result when noise is introduced is obtained with the skyline that is close to the camera. Since it changes significantly over small distances, any errors in position or distortions can complicate the task for the algorithm.

12.1.3 Test 3: Robustness to pitch and roll

The objective of these tests is to evaluate the performance of the algorithm against pitch and roll variations. These tests were performed using binary data and the extraction, transformation and estimation components were tested. Pitch and roll variations were always 10° around the nominal orientation. The panoramas used and the superimposed extracted skyline can be found in annex 2.

Id	Parameter	Error pose x (m)	Error pose y (m)	Error heading (°)	Error score
4	Pitch	265	930	173	865.88
	Roll	175	320	169	608.04
5	Pitch	365	270	154	828.15
	Roll	565	610	119	1763.86
6	Pitch	320	420	0	1949.83
	Roll	205	105	107	2280.25

Table 5: Results of the robustness to pitch and roll

The results show that the algorithm is not robust to pitch and roll variations. Thus, during on-board operation, the rover should be able to control its pitch and roll to match the values used for skyline rendering from the DEM. Otherwise, performance cannot be guaranteed. This is one of the main points to be improved. Image warping from rough pitch and roll estimates are considered for future improvement.

12.1.4 Test 4: Performance test

The whole algorithm has been evaluated using both real and simulated data in order to test its performance (precision, CPU time). The results for both acquisitions can be found in the table below. The acquisitions used and the extracted skyline superimposed to the panoramas can be found in annex 3.

Id	Type	Error pose x (m)	Error pose y (m)	Error heading (°)	Error score	Time (seconds)
7	Real	4.52	0.55	-	619.62	20.3
8	Simulated	0	0	0	16.91	36.61

Table 6: Performance results for the after-landing scenario

As can be observed, the error for the simulated data is the lowest it can be according to the chosen grid resolution for rendering. The same applies for the y position of the real acquisition. Nevertheless, the x position maintains a low error but does not match the closest position. This could be due to elements like vegetation or stones overlapping the skyline or most likely due to pitch and roll variations with respect to the virtual camera used for rendering.

Regarding the computational time, the table shows the time used by a standard computer. On a representative target processor, the time is estimated to be very approximately 30 times higher. In all cases, the time is lower than 19 minutes. Nevertheless, the size of the search area is 49 times smaller than originally planned. It is worth mentioning that most of the time taken by the algorithm is consumed by the skyline extraction step and, even if we multiplied the resulting time by 49, the time largely respects the 1-day constraint for this scenario.

A big difference in time can be observed between the real and the simulated data. This is probably due to the fact that there are a lot more images to assemble (13 vs. 5) and that the extraction is performed for a whole 360º panorama, with a lot more pixels being part of the graph.

12.2 SCENARIO 2: END OF TRAJECTORY

To test this scenario, a search grid of 20x20m with 1m resolution was considered. Only one test was performed due to the need to render new skylines for every tested position.

12.2.1 Test 5: Performance test

The whole algorithm was tested using real and simulated data. The aim of this test was to test the precision and the computational time of the algorithm. The images used and the extracted skyline superimposed to the assembled panoramas can be found in annex 3.

Id	Type	Error pose x (m)	Error pose y (m)	Error heading (°)	Error score	Time (seconds)
7	Real	5.52	0.55	-	617.07	6.69
8	Simulated	0	0	0	16.91	18.45

Table 7: Performance results for the end-of-trajectory scenario

According to the results, the algorithm is quite precise, more than expected in view of the state-of-the art. However, tests need to be performed with more data, as precision could change when the horizon is very far from the camera due to the fact that the skyline is more stable over a certain distance.

In terms of computational time, the algorithm would be able to provide an estimate in a bit over 9 minutes, so the 1-hour constraint is largely respected.

12.3 SCENARIO 3: ALONG THE TRAVERSE

The whole algorithm was tested using the same data as for the other performance tests. In this case, the normal operation of the algorithm was changed in accordance with the definition of the third scenario. Here, the algorithm takes an estimate of position as input and outputs an estimate of the rover's heading. This is done to correct the orientation every few meters of traverse. A 50x50m search area with 1m grid resolution was used.

12.3.1 Test 6: Performance test

The objective of this test was to test the precision and the computational time of the algorithm when adding error to the initial estimate. In all cases, and according to the 10° uncertainty set as requirement, the search area extends from –10 to +10° around the ground-truth orientation. As orientation was not available for real data acquired during field tests, the heading estimate obtained when setting the ground-truth position as the input of the algorithm was considered as the ground-truth orientation value (0° error). The images used and the extracted skyline superimposed to the assembled panoramas can be found in annex 3.

In the table below, δx and δy represent the errors in the input x and y positions with respect to the ground-truth, while the heading error is the output of the algorithm.

Id	Type	δx (m)	δy (m)	Error heading (°)	Error score	Time (seconds)
7	Real	0.48	0.55	0	876.71	5.39
		2.48	2.55	0	1091.05	5.75
		4.48	4.55	1	1607.06	5.65
		6.48	6.55	1	2107.97	5.72
		8.48	8.55	1	3073.03	5.89
		10.48	10.55	1	4093.54	5.54
		15.48	15.55	3	8771.46	5.38
		20.48	20.55	4	15664.5	5.75
8	Simulated	0	0	0	16.91	17.5
		2	2	0	86.47	17.98
		4	4	1	351.29	17.59
		6	6	1	646.59	17.5
		8	8	1	1323.67	16.87
		10	10	2	1983.05	16.96
		15	15	2	5574	18.44
		20	20	2	11970.7	17.03

Table 8: Performance results for the along-the-traverse scenario

The tests were made up to 20m of error in position to account for a 2% error for a 1km traverse, as described in the previous scenario. The results show that, with these conditions, the maximum 10º error is never reached. In fact, for the simulated data, the maximum orientation error is 2º, which is quite low, while for the real data the maximum is 4º. Even though this is 2 and 4 times the target value, it still seems quite moderate for a 20m error difference in position. Intuitively, we could think that this would not be the case with skylines that are closer to the camera, since they change a lot more with small camera displacements. Further testing is needed on different skyline types to verify this hypothesis. The aimed 1° error is only found with a maximum of 10m error in x and y for the real acquisition and 8m for the simulated one. However, for the real data, the values change rapidly to 3 and 4º (and we could guess that this would be the tendency), while for the simulated data they apparently change more slowly .

In terms of time, a big difference can again be observed between the real and the simulated data. If we multiply the results by 30 to account for a real demonstration scenario, the maximum is about 2.9 minutes for the real acquisitions and under 12.5 minutes for simulated data. In the second case, the result is quite over the fixed 5-minute constraint. As shown during the different tests, partial panoramas seem to work quite well in terms of performance, so this fact could be exploited to speed up the algorithm. More tests could be made to determine the minimum panorama size for which the algorithm works well.

13 FUTURE WORK

The final version of the skyline matching algorithm must be robust to any rotations of the rover's camera or imprecision on the rover's camera parameters and the whole algorithm must be optimized regarding computational time and memory usage.

A measure of uncertainty shall be implemented and flash and RAM memory requirements shall be tested. The skyline extraction component shall be evaluated using an appropriate metric.

The different priorities and tracks to be followed for each component are described next.

1) Panorama assembly:
   - Implement the different sub-functions separately and iteratively for the different images to maintain the desired image order after stitching.
   - In case stitching fails for a certain pair of images, allow stitching for the rest of the images and adjust field of view accordingly.
2) Skyline transformation: Warp the image according to pitch and roll estimates in order to add robustness to pitch and roll variations with respect to the rendered skylines.
3) Pose estimation: Add a 10° pitch and roll range to the grid parameters to allow attitude refinement and robustness to pitch and roll errors.
4) Pyramidal approach:
   - Render skyline average for each area: for each pyramidal level and grid position, render a skyline that contain all of the information of its subgrid positions present at the next level. This approach might not be appropriate if the skyline is very close to the camera and changes rapidly over small distances.
   - Use several candidates after each step: after each pyramidal step, keep several of the best-score candidates as starting grid points for the next step. This approach is definitely more robust but it increases the computational time. Testing is needed to determine the optimal number of candidates.
     Directly discard candidates that are very different: If error score is over a certain number, stop the algorithm for the particular grid position.
5) Skyline detection: Improve computational time and robustness
   - Graph using only edges and gap filling via DP: try a more traditional edge-based approach where only edge pixels are used to initialize the graph and gaps are filled using DP based on contrast, adjacencies, homogeneity, height, etc. This would highly reduce the number of pixels taken into account in the graph and thus increase computational time, though it might highly decrease robustness.
   - Reduced dense graph keeping only the 50 best scores in each column [22]. According to Ahmad et al., keeping only the 50 best score pixels in each column as nodes can highly improve computational time without degrading performance. This number could be modified after testing depending on the camera characteristics.

ETHICAL ASPECTS AND SUSTAINABLE DEVELOPMENT

On an environmental aspect, the impact related to my internship derives mainly from the energy consumption associated with the standard operation of a working office. On a daily basis, this includes the use of a computer desktop with 2 screens 5 days a week from about 9h to 18h for the whole 6-month internship. Outside this period, the computer was turned off. Other elements like lights, air conditioning and water usage compute in the overall company assessment. Commuting to and from work was always done by bike or on foot.

A particularly consuming procedure was the generation of skylines, which required the use of a special shared computer with a very powerful graphics card (Nvidia GeForce RTX 3070 Ti), which uses up to 283W at full power. The process took about 7 days at nearly full power.

The most remarkable activity in terms of CO_2 emissions was the trip to Bardenas, in Spain. A van was used to travel 1050 km round trip, in addition to several more kilometers to scout the sites inside the area. A trip like this with such a heavy vehicle and heavy equipment transportation pollutes the environment significantly. In fact, vans are responsible for around 2.5% of total EU CO_2 emissions, the actual target being 147g CO2/km. However, Bardenas was the closest site to Magellium that met the required characteristics to test my algorithm and anticipate future testing sessions for the ALPER project algorithms.

On an ethical aspect, several questions arise around the topic of space exploration, mostly related to space preservation, conservation and sustainability. As more money is invested in space and more people, as well as objects, are able to travel to space, there are several risks that come with it. One, is the potential to contaminate the ecosystems we visit, not only with the exploration itself but with the objects that are left in orbit. Another one is the risks to the astronauts, whose bodies change drastically. Should we be allowed to exploit and take unlimited resources from other planets or do we have some ethical obligations to preserve them? Is space exploration a good in itself or is it only justified for scientific purposes? What moral considerations should we apply if we discover life in another planet? This questions are hard to answer, and that is why policies and protocols are needed to guide us through space exploration in a more ethical and responsible way.

15 CONCLUSION

Globally, the objectives of the internship were reached: a skyline matching algorithm was studied, designed, implemented and evaluated. The results show that, with current developments, this technique for absolute localization can be very promising and largely surpass the precision of existing state-of-the-art solutions. Additional testing on field acquisitions is required to further characterize the algorithm.

A preliminary version of the algorithm was implemented. Further developments are still required to reach the planned final version of the algorithm. A measure of uncertainty has not yet been implemented, nor has the robustness to pitch and roll errors and memory usage was not tested and could probably be improved.

With regards to the computational time, the goal was reached for the first two scenarios but not for the third one (orientation estimation along the rover's traverse).

Many future improvements can be considered. The main priority would be to make the algorithm robust to pitch and roll errors by increasing the grid dimensions. Another option would be to modify the skyline encoding and pose estimation components and use, for instance, concavity features. For the skyline detection subcomponent, intensive testing should be performed to adjust the weight of the graph costs and set the parameters of the pre-processing step. For the pose estimation component, a pyramidal approach that uses grid subdivision and mean skylines could be conceived, as well as keeping several potential candidates after each step. A method to eliminate skylines that differ largely from the extracted one could be contemplated to speed up the algorithm.

This internship was a highly pleasant experience which has taught me many valuable lessons both technically and personally. I learned C++, a compiled programming language which is widely used in today's world, I got to use Blender, a 3D computer-graphics software with a powerful render engine. I worked with Digital Elevation Maps and QGIS and I expanded my knowledge in geospatial data management and image treatment.

I had the opportunity to be integrated in the ALPER project, a fascinating project for the ESA, and to work along a very welcoming and highly competent team. I learned a lot about space robotics, not only through my individual work but also from my co-workers. I acquired good working practices and learned agile project organization. This allowed me to develop some important skills such as time-management, prioritizing, teamwork, communication and adaptability, and it showed me the importance of team collaboration and of knowing when to ask for help.

With regards to my internship organization, I got the opportunity to work through the different phases of a typical ESA project, from the conception to the development, including the drafting of extensive technical documentation through each phase. This has provided me with a very exhaustive vision of the subject and it has greatly improved my understanding of a whole project life-cycle.

Overall, this internship has been a truly satisfying experience for me. It has provided me with the opportunity to grow professionally, and I can safely say that my understanding of the job environment, the enterprise organization and the spatial robotics industry has greatly increased. I am confident that this experience will help me with any career paths that I choose to pursue in the future.

16 BIBLIOGRAPHY

[1] "SCRUM - ECLEE | European Center for Leadership and Entrepreneurship Education," 22 01 2020. [Online]. Available: https://www.eclee.com/training/scrum/. [Accessed 06 09 2022].

[2] F. a. M. G. Stein, « Map-Based Localization Using the Panoramic Horizon, » *IEEE Transactions on Robotics and Automation,* vol. 11, n° %16, pp. 892 - 896, 1995.

[3] F. K. E. G. C. Cozman, « Outdoor Visual Position Estimation for Planetary Rovers, » *Autonomous Robots,* vol. 9, n° %12, pp. 135-150, 2000.

[4] E. E. H. J. N. G. R. W. S. M. V. M. B. Palmer, « Mercator - Independent Rover Localization Using Stereophotoclinometry and Panoramic, » *Earth and Space Science,* vol. 3, n° %112, pp. 488-509, 2016.

[5] B. R. A. F. M. K. F. Grelsson, « GPS-level accurate camera localization with HorizonNet, » *Journal of Field Robotics,* vol. 37, n° %13, 2020.

[6] E. Z. A. C. M. R. T. Z. A. Tzeng, « User-Driven Geolocation of Untagged Desert Imagery Using Digital Elevation, » chez *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2013.

[7] L. Č. M. E. E. S. H. Baboud, « Automatic Photo-to-Terrain Alignment, » chez *Baboud, L., Čadík, M., Eisemann, E., Seidel, H.*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2011.

[8] Z. T. J. T. T. X. X. W. Z. Pan, « Camera Geolocation Using Digital Elevation Models, » *Applied Sciences,* vol. 10, n° %119, 2020.

[9] B. Nagy, « A New Method of Improving the Azimuth in Mountainous Terrain, » *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science volume ,* vol. 88, pp. 121 - 131, 2020.

[10] C. D. J. T. A. a. S. A. Rodin, « Skyline Based Camera Attitude Estimation, » *2018 IEEE 15th International Workshop on Advanced Motion Control (AMC),* pp. 306-313, 2018.

[11] G. S. O. K. K. P. M. Baatz, « Large Scale Visual Geo-Localization of Images in Mountainous Terrain, » chez *Proceedings of the 12th European conference on Computer Vision - Volume Part II*, 2012.

[12] A. V. B. X. E. L. K. T. H. E. R. J. D. M. B. G. F. T. Nefian, « Planetary rover localization within orbital maps, » chez *2014 IEEE International Conference on Image Processing (ICIP)*, 2014.

[13] S. P. M. S. ,. M. ,. M. L. M. G. Zhu, « Skyline Matching: A robust registration method between Video and GIS, » chez *Usage, Usability, and Utility of 3D City Models*, 2012.

[14] M. C. M.-Y. L. C.-C. S. A. Fang, « Skyline for video-based virtual rail for vehicle, » chez *Proc. IEEE International Sympos. On Intelligent Vehicles.*, 1993.

[15] W.-N. L. T. C.-I. L. T.-C. H. K.-S. Lie, « A robust dynamic programming algorithm to extract skyline in images for navigation, » *Pattern Recognition Letters,* vol. 26, n° %12, pp. 221-230, 2005.

[16] B.-J. S. J.-J. N. H.-J. K. J.-S. Kim, « Skyline Extraction using a Multistage Edge Filtering, » *World Academy of Science, Engineering and Technology, International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering,* vol. 5, pp. 787-791, 2011.

[17] M. S. L. S. M. M. S. A. C. Ayadi, « Parametric Algorithm for Skyline Extraction, » chez *Advanced Concepts for Intelligent Vision Systems: 17th International Conference*, 2016.

[18] S. K. I.-C. K. J. S. Yang, « Robust skyline extraction algorithm for mountainous images, » chez *VISAPP 2007 - 2nd International Conference on Computer Vision Theory and Applications*, 2007.

[19] J. K. S.-O. K. G. S. K. I.-C. Woo, « Robust Horizon and Peak Extraction for Vision-based Navigation, » chez *Proceedings of the IAPR Conference on Machine Vision Applications (IAPR MVA 2005)*, 2005.

[20] Y. -F. K. D. L. J. R. Z.-u. Shen, « A Hierarchical Horizon Detection Algorithm, » *EEE Geoscience and Remote Sensing Letters,* vol. 10, n° %11, pp. 111-114, 2013.

[21] O. B. G. K. K. L. L. P. M. Saurer, « Image Based Geo-localization in the Alps, » *International Journal of Computer Vision,* vol. 116, n° %13, pp. 213-225, 2015.

[22] T. B. G. N. M. N. N. A. V. F. T. Ahmad, « An Edge-Less Approach to Horizon Line Detection, » chez *14th International Conference on Machine Learning and Applications*, 2015.

[23] T. B. G. R. E. N. A. V. Ahmad, « A Machine Learning Approach to Horizon Line Detection using Local Features, » chez *Proceedings of 9th International Symposium on Visual Computing (ISVC)*, 2013.

[24] T. B. G. N. M. N. N. A. V. F. T. Ahmad, « Horizon line detection using supervised learning and edge cues, » *Computer Vision and Image Understanding,* vol. 191, n° %14, 2020.

[25] J. S. E. D. T. Long, « Fully Convolutional Networks for Semantic Segmentation, » chez *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[26] V. K. A. C. R. Badrinarayanan, « SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation, » *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 39, n° %112, pp. 2481-2495, 2017.

[27] D. F. P. T. R. N. Frajberg, « Convolutional Neural Network for Pixel-Wise Skyline Detection, » chez *International Conference on Artificial Neural Networks*, 2017.

[28] F. M. Y. T. J. H. Y. Z. L. Guo, « Robust and Automatic Skyline Detection Algorithm Based on MSSDN, » *Journal of Advanced Computational Intelligence and Intelligent Informatics,* vol. 24, n° %16, pp. 750-762, 2020.

[29] T. B. G. R. E. N. A. V. F. T. Ahmad, « An Experimental Evaluation of Different Features and Nodal Costs for Horizon Line Detection, » chez *10th International Symposium on Visual Computing (ISVC)*, 2014.

[30] T. C. P. Č. M. B. G. Ahmad, « Detection, Comparison of Semantic Segmentation Approaches for Horizon/Sky Line, » chez *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017.

[31] T. E. E. Č. M. B. G. Ahmad, « Resource Efficient Mountainous Skyline Extraction using Shallow Learning, » chez *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021.

[32] P. C. P. B. T. D. Furgale, « A Comparison of Global Localization Algorithms for Planetary Exploration,» chez *The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[33] S. M. L. P. M. M. G. Zhu, « Video/GIS registration system based on skyline matching method,» chez *IEEE International Conference on Image Processing*, 2013.

[34] J. H. T. Ventura, « Structure and motion in urban environments using upright panoramas,» *Virtual Reality,* vol. 17, n° %12, pp. 147-156, 2013.

[35] B. F. M. I. F. Grelsson, « Highly Accurate Attitude Estimation via Horizon,» *Journal of Field Robotics,* vol. 33, n° %17, pp. 967-993, 2016.

[36] F. a. K. E. Cozman, « Automatic Mountain Detection and Pose Estimation for Teleoperation of Lunar Rovers,» chez *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, 1997.

[37] L. L. S. Wei, « 3D Peak Based Long Range Rover Localization,» chez *2016 7th International Conference on Mechanical and Aerospace Engineering*, 2016.

[38] Y. Q. G. G. K. G. H. S. K. C. Chen, « Camera geolocation from mountain images,» chez *Proceedings of the 18th International Conference on Information Fusion*, 2015.

[39] A. E. J. B. D. Nuchter, « Skyline-based registration of 3D laser scans,» *Geo-spatial Information Science,* vol. 14, n° %12, pp. 85-90, 2011.

[40] D. S. B. J. R. D. B. A. Y. Y. M. Bolme, « Visual object tracking using adaptive correlation filters, » chez *roceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.

[41] S. P. M. D. S. B. L. Chiodini, « Mars rovers localization by matching local horizon to surface digital elevation models,» chez *2017 IEEE International Workshop on Metrology for AeroSpace* , 2017.

[42] P. W. D. S. J. Gibbens, « Efficient Terrain-Aided Visual Horizon Based Attitude Estimation and Localization,» *Journal of Intelligent & Robotic Systems,* vol. 78, n° %12, 2014.

[43] M. V. L. S. M. M. S. A. C. Ayadi, « The skyline as a marker for augmented reality in urban context,» chez *13th International Symposium on Visual Computing*, 2018.

[44] V. B. S. N. Gupta, « Terrain-based vehicle orientation estimation combining vision and inertial measurements,» *Journal of Field Robotics,* vol. 25, n° %13, pp. 181-202, 2008.

[45] A. Rosebrook, "Image Stitching with OpenCV and Python," PyImageSearch, 17 12 2018. [Online]. Available: https://pyimagesearch.com/2018/12/17/image-stitching-with-opencv-and-python/. [Accessed 01 09 2022].

17 ANNEXES

17.1 ANNEX 1: SKYLINE EXTRACTION WITH DIFFERENT HORIZON TYPES AND NOISE LEVELS

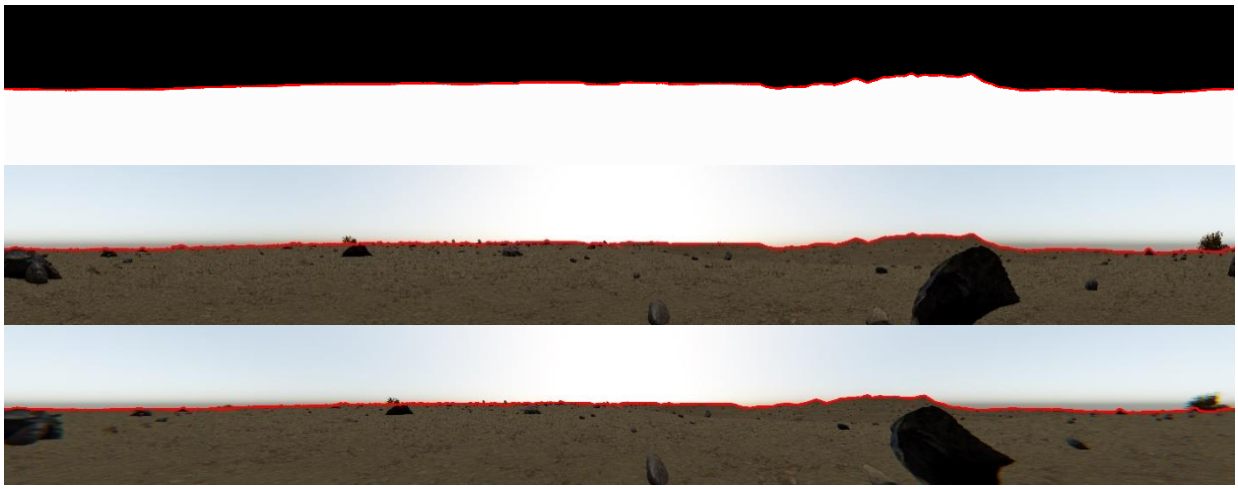


Figure 27: Extracted skyline on acquisition id 1, noise level 1, 2 and 3 (from top to bottom)

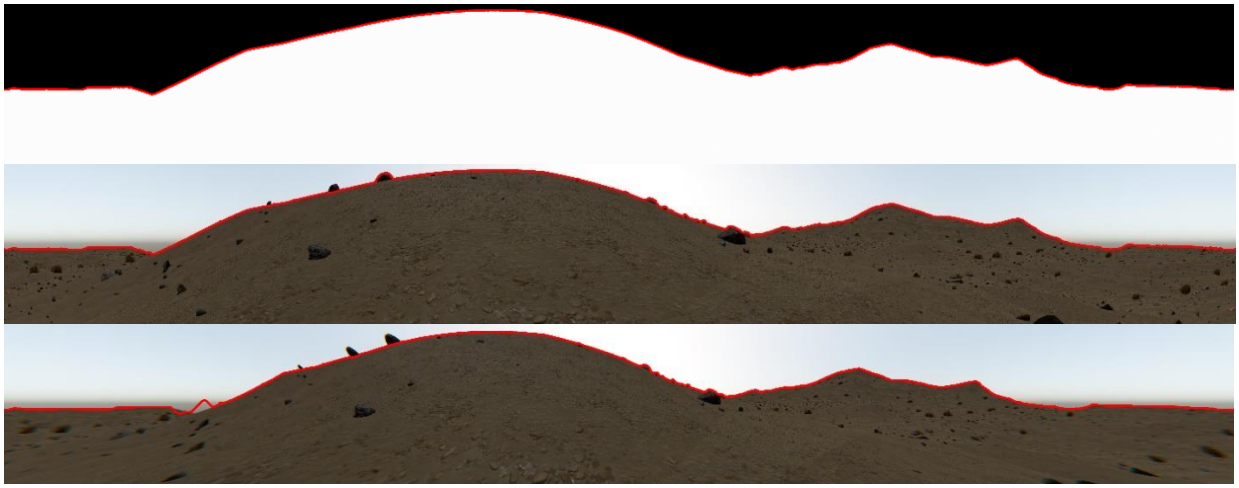
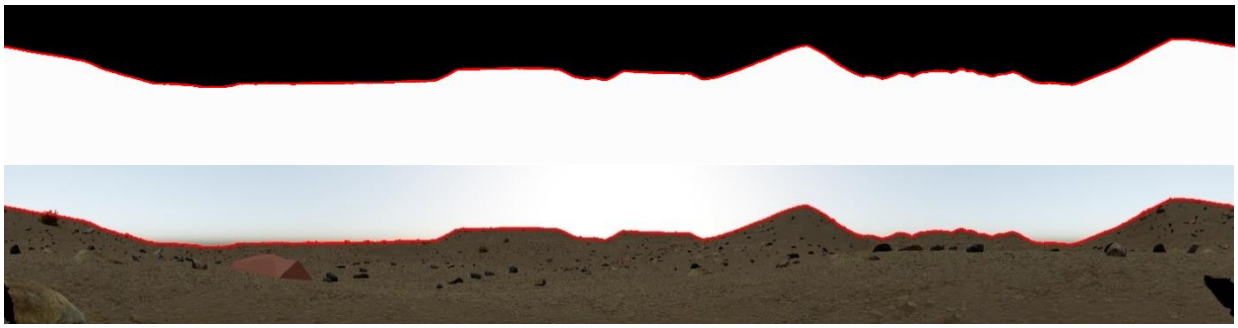


Figure 28: Extracted skyline on acquisition id 2, noise level 1, 2 and 3 (from top to bottom)



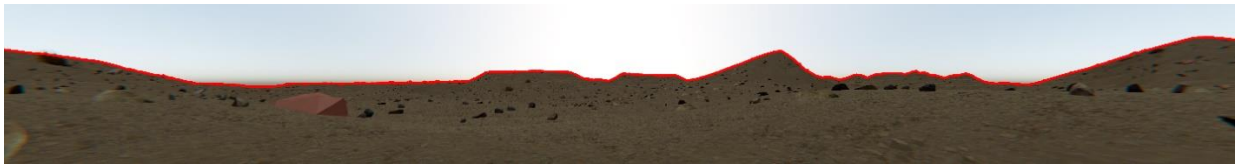


Figure 29: Extracted skyline on acquisition id 3, noise level 1, 2 and 3 (from top to bottom)

17.2 ANNEX 2: SKYLINE EXTRACTION WITH PITCH AND ROLL ERRORS

Figure 30: Extracted skyline on acquisition id 4, pitch (1st row), roll (2nd row) and panorama at 0 pitch and roll (3rd row)

Figure 31: Extracted skyline on acquisition id 5, pitch (1st row), roll (2nd row) and panorama at 0 pitch and roll (3rd row)

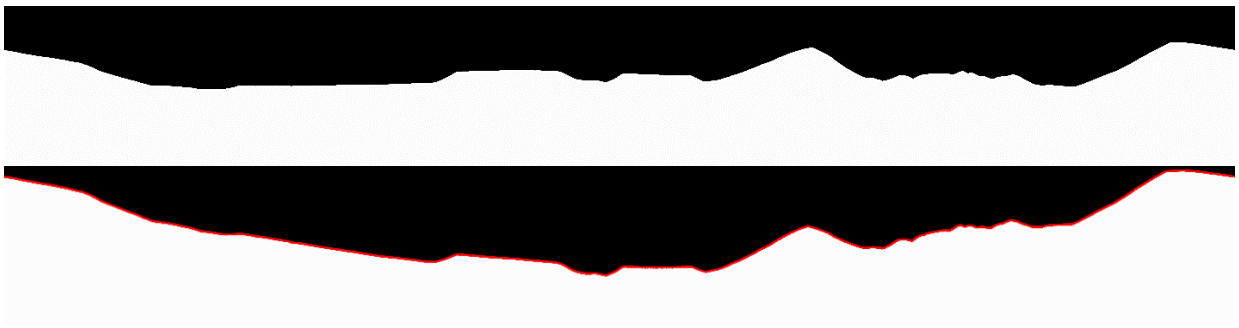


Figure 32: Extracted skyline on acquisition id 6, pitch (1st row), roll (2nd row) and panorama at 0 pitch and roll (3rd row)

17.3 ANNEX 3: IMAGES AND SKYLINE EXTRACTION FOR THE PERFORMANCE TESTS

Figure 33: Acquisition id 7, images

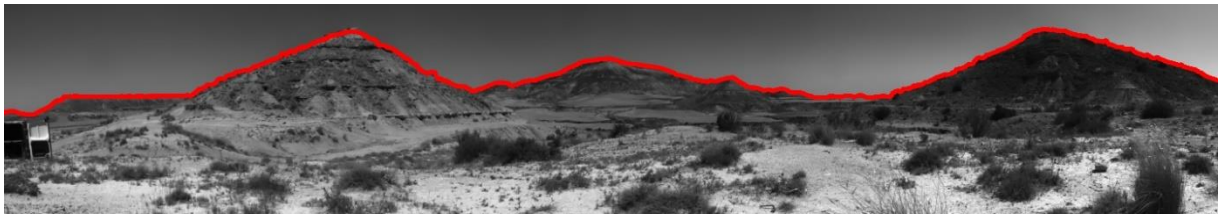


Figure 34: Acquisition id 7, extracted skyline

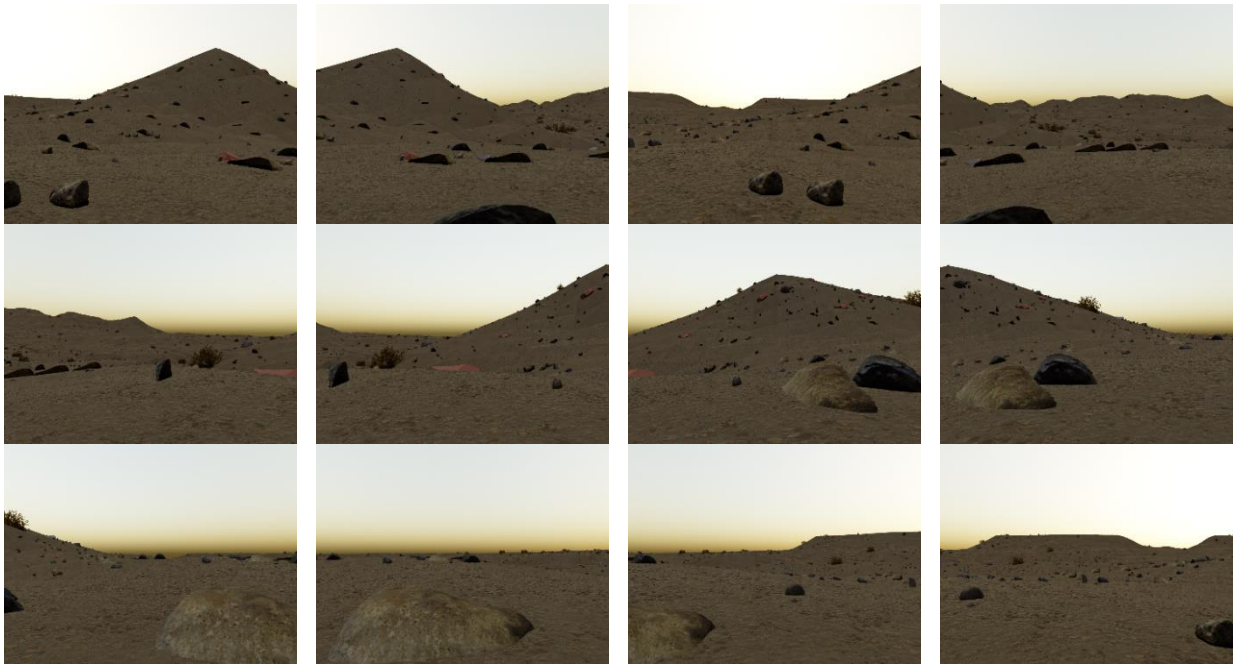


Figure 35: Acquisition id 8, simulated acquisitions

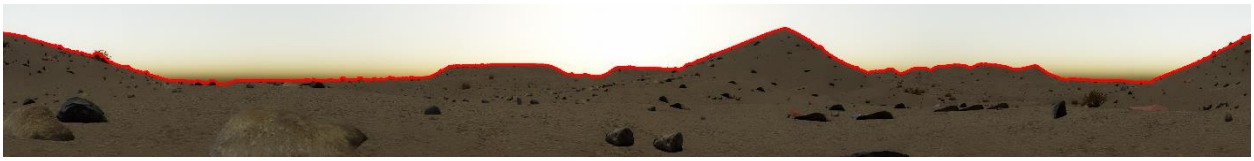


Figure 36: Acquisition id 8, extracted skyline

17.4 ANNEX 4: MAGELLIUM'S TEST ROVER STEREO-BENCH SPECIFICATIONS

	Magellium	ExoMars NavCam	Mars 2020 NavCam
Baseline	Adjustable : 12cm to 22cm (can be extended)	15cm	42.4cm
Boresight mounting orientation	Mounted to PTU, Left/right camera boresights are parallel	Mounted to PTU, Left/right camera boresights are parallel	Mounted to PTU, Left/right camera boresights are parallel
Pixel format	1600*1200	1024*1024	5120*3840
Horizontal FOV	62°	65°	96°
Vertical FOV	48°	35°	73°
Focal length	6mm	4mm	19.1mm
Aperture	f/1.4	f/8	f/12
Height above surface	75cm (can be extended)	205cm	198cm
Angular range of the PTU	Horizontal: 360° Vertical: 180°	Horizontal: 360° Vertical: 180°	Horizontal: 360° Vertical: 180°

Table 9: Magellium's stereo-bench specifications