POLITECNICO DI TORINO

Master's Degree in Software Engineering

Master's Degree Thesis

Matching Algorithms for Target Localization in Pinpoint Landing

Supervisors

Prof. Andrea BOTTINO

Ing. Paolo MARTELLA

Dott. Barbara ACQUAROLI

Candidate

Riccardo VUKOVIC

April 2025

Abstract

Pinpoint landing identifies the capability to land on a planet, e.g. Mars or the Moon, with a position error lower than few tens of meters. To reach this challenging aim different technologies are involved. Looking deeper at the area of the terrain relative navigation and strategies, this work focuses in particular on the development and analysis of two matching algorithms designed to initialize the landing sequence. The primary purpose of the matching algorithms is to provide an initial estimate of the spacecraft pose with respect to the landing target, encompassing both position and attitude. This estimation serves as a crucial input for initializing the guidance and navigation system during the final descent phase towards the established landmark.

To achieve this, during the descent phase a Landing Vision System (LVS) mounted on the spacecraft captures multiple images of the planetary surface and various landmark features - such as radius and relative coordinates - are extracted. The matching algorithm then compares these obtained features with a preloaded map of a broader region encompassing the captured image.

For the objective of this dissertation, two matching algorithms have been developed and analyzed. The first algorithm is based on the construction of triangular configurations, leveraging the geometric similarity of these structures for feature matching, whilst the second one seeks the transformation that optimally aligns the extracted features with those present in the reference map. Furthermore, a Monte Carlo analysis was conducted to assess the performance of the aforementioned methods, accounting for errors affecting both attitude and altitude. The results demonstrate that both algorithms exhibit robustness, even under challenging conditions. While the first algorithm is more resilient to errors, it entails a higher computational cost. Conversely, the second algorithm offers significantly improved speed but exhibits greater susceptibility to inaccuracies and errors.

In the conclusions, this thesis describes the manners and constraints for the applicability of such typologies of algorithms in the frame of the future exploration missions, manned and unmanned, where position accuracy at landing will become more and more crucial.

Acknowledgements

Desidero ringraziare la mia famiglia per il prezioso supporto ricevuto lungo tutto il percorso. Un ringraziamento speciale va a Dorothy, il cui sostegno, anche nei momenti più complicati, è stato fondamentale per il raggiungimento di questo traguardo.

Vorrei ringraziare anche il mio relatore e i tutor di Thales Alenia Space, in particolare Paolo, per la sua disponibilità e competenza, indispensabili per la realizzazione di questo lavoro.

Table of Contents

List of Tables

List of Figures

Acronyms

LVS

Landing Vision System

LiDAR

Light Detection and Ranging

RDA

Radar Doppler Altimeter

PDS

Parachute Deployment System

TSM

Triangle Similarity Matching

RANSAC

Random Sample Consensus

MC

Monte Carlo

LTF

Landing Target Frame

CCD

Charge-Coupled Device

Chapter 1

Introduction

The exploration of the Moon and Mars has continuously fascinated humanity, inspiring different kinds of art and disciplines, before becoming the focus of scientists and engineers. The Soviet Union pioneered human spaceflight by sending the first astronaut into orbit in 1961, while the United States achieved the first human landing on the Moon in 1969 through the Apollo program. However, the first successful mission to Mars was Mariner 4 in 1965 from NASA, which provided the first close-up images of the Red Planet. The Soviet Mars 3 mission in 1971 became the first spacecraft to achieve a soft landing on Mars, though it failed shortly after touchdown. Viking 1 and Viking 2 missions from NASA in 1976 successfully landed and transmitted scientific data, marking a significant milestone in Mars exploration.

Following the Moon landing, governmental interest in space exploration declined, but renewed attention emerged towards the end of the 20th century. NASA resumed Mars missions, deploying numerous orbiters, landers, and rovers.

Mars landings require different strategies depending on the precision needed. Ballistic entry is suitable for missions with less stringent accuracy requirements, with landing zones spanning up to 50 km due to atmospheric variations. For missions requiring precision within a few kilometers, guided entry systems actively control descent by modulating drag and lift. Future missions demanding pinpoint accuracy within 100 meters will require advanced technologies, including optimized guidance, terrain navigation, and obstacle avoidance.

During the final descent phase, parachutes deploy, and the thermal protection

shield is released, exposing the radar altimeter and Landing Vision System (LVS). The LVS is crucial for pinpoint landings, as radar alone can determine altitude and velocity but lacks precise positional accuracy. The LVS captures images of the Martian terrain, compares landmark features with preloaded maps, and allows real-time location estimation. The process initiates at an altitude of approximately 4 kilometers. During the image processing and feature-matching phase, the lander, descending under parachute, maintains a velocity of around 100 m/s. Prior to reaching an altitude of approximately 2,500 meters, the estimated position of the lander relative to the designated landing target is provided to the optimized guidance system. This system computes the trajectory towards the target while considering kinematic, dynamic, and operational constraints.

Upon completing this phase, the lander transitions to the controlled descent phase. A maneuver is commanded to reorient the longitudinal axis of the lander to a prescribed off-vertical attitude. Subsequently, a divert maneuver is executed to correct the horizontal displacement and align the vehicle above the landing site. Finally, a braking maneuver is performed to achieve a soft landing in proximity to the target location.

Terrain navigation continuously tracks landmarks in the LVS field of view, using cameras or LiDAR to update the position and orientation of the lander. In cases where onboard sensors detect unexpected hazards, last-minute obstacle avoidance maneuvers can be executed to ensure a safe landing.

This thesis focuses mainly on the features comparing process, developing two different algorithms along with the whole descent environment model, suited for the Martian specific setting, to simulate as accurately as possible a realistic scenario for future missions.

The following dissertation is structured to offer a thorough exploration of the problem and the solutions devised. Chapters 2 and 3 provide an overview of the Pinpoint Landing Problem, analyzing objectives and constraints and depicting the state of the art for the matching algorithms. Chapter 4 and 5 present the methodology adopted to create the simulation environment. Chapter 6 focuses on the matching algorithms and their different approaches while an experimental analysis is proposed in Chapter 7, where the results of the two methods are compared in different case scenarios and proposing future research directions. Finally, Chapter 8 concludes the thesis by summarizing the results.

Chapter 2

Problem Overview

In this prefatory chapter there will be a more detailed overview of the pinpoint landing problem, with particular attention to the absolute localization and its challenges, delving into the processes of landmark detection and the obtained features, proceeding with the core of the study, the matching algorithms, and the constraints that have to be taken into account.

2.1 Pinpoint Landing problem

As previously mentioned, the pinpoint landing is a fundamental operation for the purposes of upcoming future missions in the field of space exploration. The capability to land in a precise location, with less than 100 meters of error, allows for the allocation of resources in specific outpost, avoiding hazard regions, and enabling the exploration of highly specified areas otherwise difficult to reach.

At the end of the entry phase a parachute or a number of successive parachutes are opened, and successively the thermal protection shield is released enabling the Radar Doppler Altimeter (RDA) and the Landing Vision System to look at the soil. For both Lunar and Martian missions, when pinpoint landing is required, the Landing Vision System is fundamental because the radar may be only able to determine the altitude or the velocity vector (in case the Doppler channel is integrated) but not the latitude and longitude distance from the target. By using the LVS an image of the surface can be captured, landmark features may be identified and compared with the ones in a map, including the observed area, preloaded in the

onboard computer database before launch. The activity is fundamental to initiate the whole sequence of pinpoint landing and is actually the objective of this thesis.

Indeed, the identification of observed terrain features within the image enables the quantification of positional error associated with the current position of the lander, with respect to the landing target, when the altitude is still about 4 km above the planet surface. The lander in this phase is still hanged to the parachute limiting its vertical velocity. The computer can so take a certain number of seconds (order of one decade) to elaborate the images and accomplish the mentioned task of initial localization. The successive task is represented by the elaboration of a kinematic and dynamic profile to be followed by the control (starting at an altitude of about 2 km). The task exploits optimized algorithms to identify, at first, the most convenient flight duration to the target and then a sequence of commands allowing to simultaneously fulfil all the kinematics, dynamic and operational constraints of the landing.

During this phase the terrain navigation is in charge to track the landmarks changing in the field of view of the LVS (that could be based on cameras or LiDAR) so providing the updated lander state (translation and rotation terms) time by time, whilst the navigation could be on call for a last occurrence task of obstacle avoidance in case the proximity observation of the surface is detecting some unexpected hazards not evident from the remote observation.

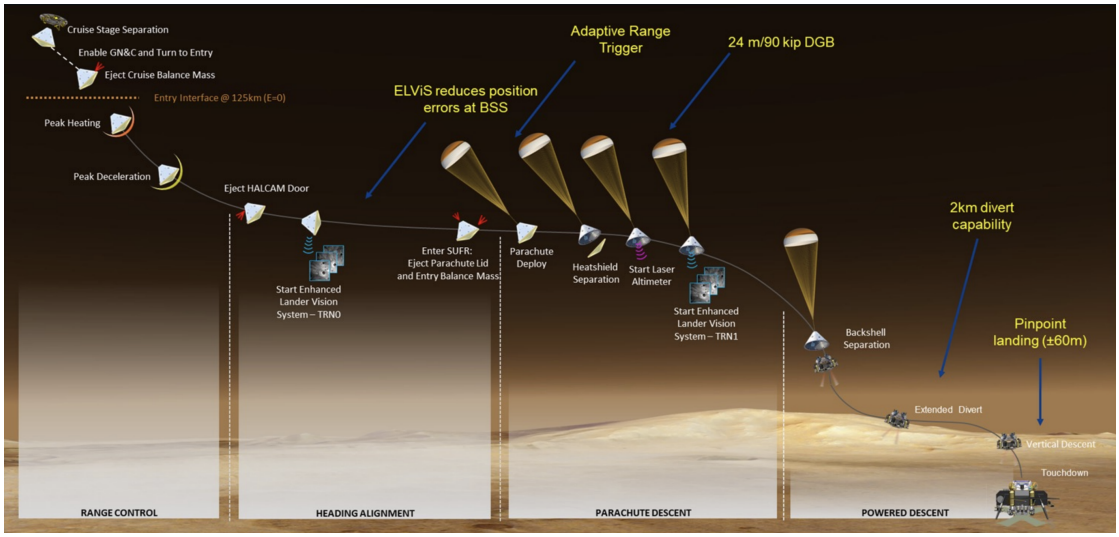

Figure 2.1: Pinpoint landing scheme, example from MSL mission [1]

2.2 Landmark Detection and Features Extraction

Landmarks, especially in the context of Mars exploration, can be of different types, such as craters, rocks, ridges, depressions, and other geological features that can be used to identify a specific area. Typically, since they often recall a circular or elliptic shape, it is useful to consider landmarks as circumferences to simplify the detection process and yet provide significant characteristics.

Nowadays, landmark detection and feature extraction are predominantly performed using conventional methods, though deep learning-based approaches are increasingly being adopted due to rapid advancements in the field in recent years. Classical algorithms utilize image processing approaches to identify landmarks, leveraging on edge-to-edge detection, Hough transforms, template matching methods, watershed algorithms and threshold segmentation [2]. These traditional techniques can be effective on small datasets and in scenarios where the landmarks are well-defined and easily distinguishable, but they can struggle as the dataset grows, illumination noise increases, or asset and altitude errors are present, leading to scaling and perspective errors. On the other hand, deep learning algorithms can be trained on large landmark datasets of the desired surface and can provide a more robust and accurate detection, even in the presence of sustained noise, leveraging object and pattern recognition, semantic segmentation and classification tasks. However, they require detailed and well-annotated datasets that are not always available, and they can be computationally expensive and time-consuming, thus not always suitable for the technology constraints of the on-board computers and time requirements. For the purpose of this thesis, a classical approach is presented, which is more suitable for the size and characteristics of the synthetic map generated for the study.

2.3 Matching Algorithm

For the successful identification of the spacecraft location inside the observed area, the extracted features obtained from the corresponding landmarks captured in the LVS image have to be compared to the ones already stored in an on-board database of known landmarks of the same—but broader— area. However, the comparison is not straightforward due to the presence of different sources of noise,

such as perturbations in the spacecraft pose and altitude. Albeit it is safe to assume that the spacecraft entry phase is deterministic, that is the vehicle pose, altitude and velocity are known with a high degree of precision, minor errors can affect these estimated values, leading to slight rotations along the three axes, entailing rotation and perspective differences, and altitude noise that is responsible for an additional scaling factor to be taken into account. Thus, the matching algorithm has to tackle these issues and provide a robust and accurate estimate of the spacecraft coordinates, avoiding mismatching and false positives that could lead to a wrong localization and consequently to a possible failure of the landing. Since the algorithm has to provide a result in the least amount of time possible (in the order of a decade of seconds), it is crucial that it is computationally efficient and able to run on a low-power device such as the on-board computer of the spacecraft, and it's also fundamental that it can always produce a successful result, whether it is reliable or not. In this dissertation, two different algorithms are proposed, each one with its own strengths and weaknesses in terms of computational time, accuracy and success rate, and they are compared to each other but also evaluated independently, in order to provide a comprehensive analysis of their performances in different scenarios.

2.4 Constraints and Output Target

For the purpose of this study, the following constraints on errors have to be taken into account, considering the on-board instrumentation:

Actual Parameters

- Altitude: 4100 m ± 65 m
- Position:
  - x: 0 ± 3000 m
  - y: 0 ± 3000 m
- Pose:
  - Off-vertical Axes: 0 ± 5 degrees
  - Perpendicular Axis: 0 ± 180 degrees

Estimated Parameters

- Altitude: 4100 m
- Pose:
  - Off-vertical Axes: 0 ± 1 degree
  - Perpendicular Axis: 0 ± 1 degree

Target

The target is represented by the ultimately estimated coordinates of the spacecraft position:

- Latitude
- Longitude

Chapter 3

A Review of Existing Landmark Matching Algorithms

While landmark matching for precision landing represents a relatively nascent area of research, a considerable number of methodologies have been developed to address this challenge. Over recent years, this field has experienced substantial growth, with significant investments from various space agencies aimed at creating sophisticated technologies to enhance landing accuracy.

Leroy et al. pioneered an absolute navigation technique utilizing craters, initially conceived for an asteroid landing mission [3]. Their approach involved detecting craters through the identification of their perimeters and subsequent fitting of ellipses. A pre-existing crater catalog served as the reference dataset against which the detected craters were projected and matched. However, this method was limited to resolving the affine transformation between the two crater sets, thus requiring a reasonably accurate initial estimate of the spacecraft's pose.

Chang et al. further explored crater-based absolute navigation through several iterations, initially focusing on asteroid landings before adapting their work for a Mars mission. Their methodology employed edge detection to identify craters within lander imagery and subsequently matched these observed craters with known ones by utilizing co-planar invariants derived from the crater ellipses [4].

Singh and Lim proposed a method specifically designed for fully autonomous absolute navigation, eliminating the need for prior pose information, except for the requirement for a nadir-pointing navigation camera [5]. Their work marked the first instance of crater detection through image segmentation, as opposed to traditional edge detection, although it relied on fixed thresholds for this process.

Similarly, the method developed by Hanak and Crain also necessitated a nadir-pointing camera for spacecraft self-localization [6]. This requirement stemmed from their crater candidate extraction technique, which employed circular edge curve detection. For crater matching, they utilized a k-vector approach, analogous to those used in star trackers, which was further augmented with an additional crater to enhance robustness. Their method demonstrated global lunar localization within a defined altitude range.

An extensive study by Simard Bilodeau refined crater detection by integrating segmentation with an edge-based approach to improve resilience against erroneous detections [7]. However, the crater matching component of this method relied on an initial pose estimate provided by an external navigation filter. This filter was assumed to be operational, projecting the crater catalog onto the image plane to facilitate the matching of detected craters. Consequently, a successful initialization of the system was deemed necessary to obtain a sufficiently accurate pose estimate.

Maass et al. defined a methodology that integrated three distinct crater matching algorithms, each tailored to different phases of vehicle descent, addressing scenarios ranging from a lost-in-space situation to one with an estimated initial pose [8]. Their approach was designed to be robust against false detections and capable of functioning in a fully autonomous manner.

Jin et al. introduced a strategy to mitigate challenges associated with deep learning-based crater detection and matching by fusing geometric and regional descriptors, thereby providing a robust solution for visual navigation tasks [9].

A novel technique was presented by Uwano et al. [10], which leverages triangle geometric features to determine the similarity between landmark triads observed by the camera and those within a preloaded database.

Finally, another notable approach was developed by Li et al. [11], which utilizes the centers and radii of landmarks to construct feature histogram descriptors for preliminary matching, followed by the application of Random Sample Consensus (RANSAC) to eliminate spurious matches.

The latter two methodologies will be examined in greater detail in subsequent chapters, as the algorithms proposed in this dissertation draw inspiration from them.

Chapter 4

Simulated Environment

To properly face the pinpoint landing problem is necessary to provide an environment that can handle the simulation of the spacecraft descent, resembling as more accurately as possible real terrain characteristics. To do so, the PANGU tool played a relevant role in the creation of a synthetic landmark map that has been used to test the simulations of this thesis. In this chapter a brief introduction to PANGU will be given, outlining its main features and capabilities, followed by the methods used to generate the actual dataset starting from the tool output.

4.1 PANGU Tool

PANGU (Planet and Asteroid Natural scene Generation Utility) is a simulation tool designed to validate autonomous spacecraft navigation and landing systems [12]. It generates high-resolution synthetic imagery and LiDAR data, replicating planetary or asteroid surfaces with multi-resolution terrain modeling. The software enables the creation of synthetic real-world terrains, incorporating features like craters and boulders to simulate hazardous landscapes. It also models small celestial bodies, allowing users to import existing shapes or generate synthetic asteroids with realistic surface textures.

PANGU includes customizable camera and LiDAR simulations, emulating optical distortions, sensor noise, and motion effects, rendered via GPU for real-time performance. Dynamic mission phases, such as landing sequences, can be visualized using FFmpeg-generated videos. Validated against data from missions like

Hayabusa, PANGU has been refined over two decades and serves as a critical tool for European space exploration, aiding navigation algorithm development and hazard avoidance testing.

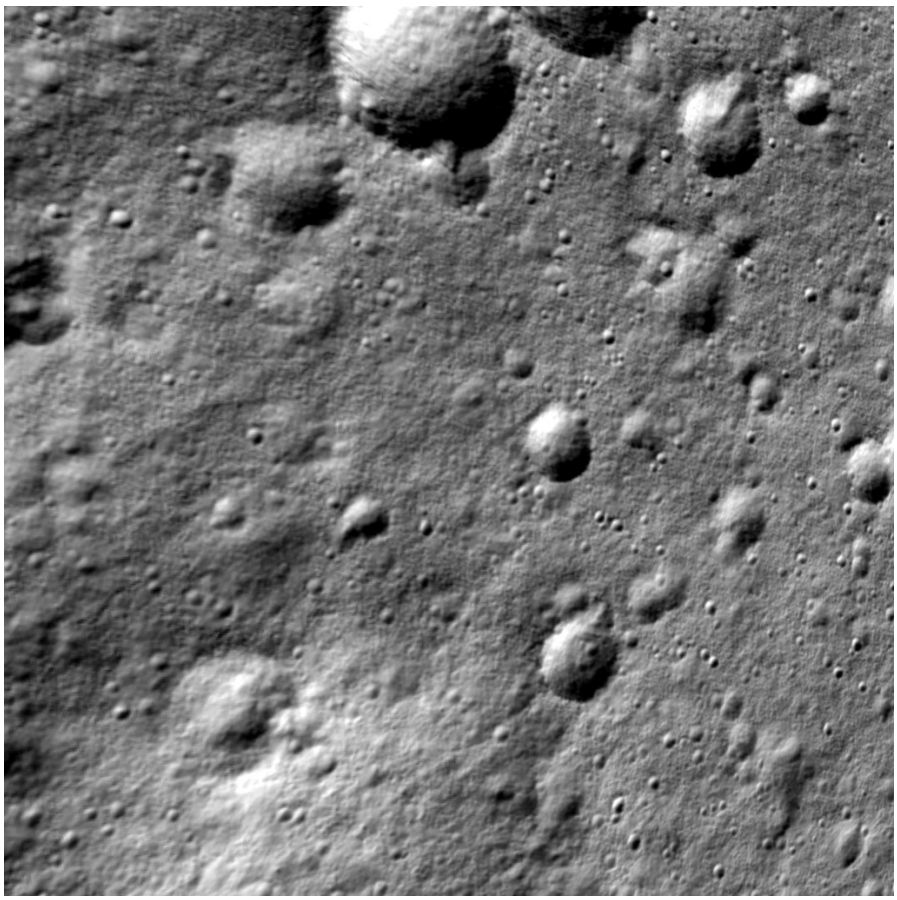

Figure 4.1: Example of PANGU generated terrain image

4.2 Synthetic Terrain Map Generation

For the specific case of this dissertation, the PANGU tool has been used to generate a synthetic terrain map of a Martian surface, providing more than 100 images of a reference area.

Starting from the data generated by the software, salient features of the landmarks inside the area have been extracted, considering them as circles. In particular,

latitude and longitude coordinates along with radius and age were taken into account. Since PANGU provides a real-time render of the scene, extracted features were taken at diverse times and heights during the descent, creating a heterogeneous landmark dataset encompassing a 256 km^2 surface area divided in 5 progressively decreasing size frames of 16x16, 8x8, 4x4, 2x2, 1x1 km^2 (according to the elevation reduction).

To achieve a high uniformity in terms of landmark characteristics, age and extension have been considered, where the first value defines how much the edge has been flattened over time and thus determining the capability to be identified by the LVS. A normal distribution has been applied and entries with an age outside the range of [50 - 90] years and a diameter $>$ 300 m were filtered out.
Afterward, the 5 different frames of the scene just filtered have been merged, resulting in a total of 2529 landmarks uniformly distributed over the 256 km^2 surface (Figure 4.2).

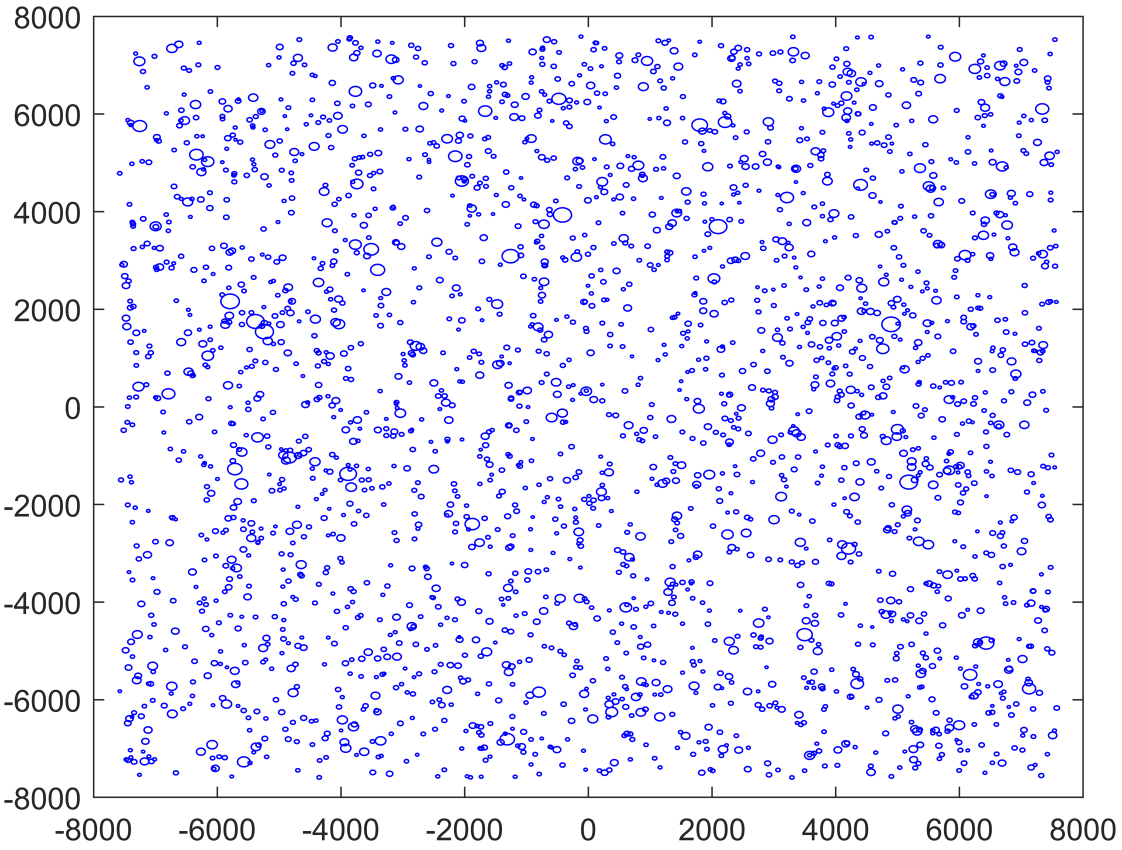

Figure 4.2: Synthetic landmark map

Chapter 5

Landmark Detection and Feature Extraction

In order to implement the landmark detection process, a real-time model of the landing vision system has been developed. It is implemented using the Simulink framework and exploiting the pinhole model. The simulation system mimics the descent of the spacecraft from 4100 m of altitude, that is the elevation at which the Radar Doppler Altimeter begins his function, to an altitude of about 2400 m, where the Terrain Relative Navigation system is provided with the estimated position and is able to start the tracking phase towards the landing target.

The model is designed to refresh its state variables at a rate of 10 Hz, providing different captured images of the terrain during the process. Notwithstanding, only the first shot (at time $t = 0$ s) is analyzed for each simulation, as it already provides enough insights for this study purposes.

5.1 Pinhole Model

The Pinhole Model is based on a sequence of transformations enabling the representation of a solid 3-Dimensional object in the 2-Dimensional plane of the image.

Firstly, a point on the planetary surface p_M can be transformed into the camera frame p_C. Both p_M and p_C are 3-Dimensional vectors at this level. The transformations involved in the model are defined as follows:

$$\mathbf{p}_{i,C} = A\left[\mathbf{q}_{M_k}^{B}\mathbf{q}_{B}^{C}\right]\left(\mathbf{p}_{i,M} - \mathbf{r}_{M_k} - A\left[\mathbf{q}_{M_k}^{B}\right]^{T}\mathbf{r}_{C,B}\right)$$

- $\mathbf{q}_{M,k}^{B}$ is the quaternion at time k from the Landing Target Frame (LTF) to the body frame.
- $\mathbf{r}_{M,k}$ is the distance between surface and the lander at time k.
- $\mathbf{r}_{C,B}$ is the fixed distance between the body frame and the camera.

The following relationship produces the 2D image of the object. A point in the camera frame p_C can be transformed into a point of the camera image p_P through:

$$\boldsymbol{p}_{i,P} = \begin{bmatrix} h/2 \\ w/2 \end{bmatrix} - \frac{f}{\boldsymbol{e}_3^T \boldsymbol{p}_{i,C}} \begin{bmatrix} \boldsymbol{e}_2^T \\ \boldsymbol{e}_1^T \end{bmatrix} \boldsymbol{p}_{i,C} \tag{5.1}$$

Where:

- w, h are the width and height of the charge-coupled device camera (CCD).
- $f = 13.5$ mm is the focal length (from which the half-cone aperture of $\theta \approx 30$ deg is derived (5.2)).
- For the adopted camera, $w = h = 11$ mm.

$$\theta = \tan^{-1}\left(\frac{\sqrt{h^2 + w^2}}{2f}\right) \tag{5.2}$$

The transformation of the images from the camera images to the linked objects observed on the ground is therefore possible by inverting the process of the roto-translation transformations described above.

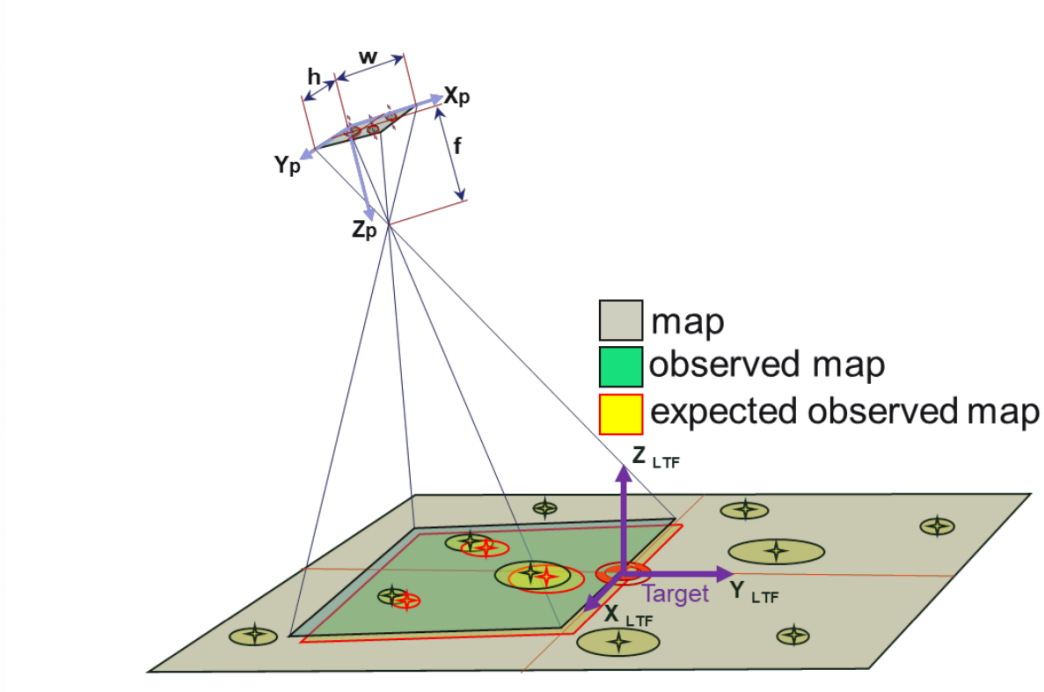

Figure 5.1: Pictorial representation of the pinhole model of the camera. The Landing Target Frame (LTF) is referenced as the East-North-Up (ENU) reference frame.

5.2 Simulink Environment

The Simulink environment above described is structured in a hierarchical manner, with the top-level diagram presenting 3 main blocks, as described in Figure 5.2:

- **Input of the Actual Pose**
  This block provides the actual pose of the spacecraft, i.e. the real position and attitude of the spacecraft. These input values establish in which position the spacecraft is at the beginning of the simulation and its orientation.

- **Ground-to-Camera Transformation**
  The following unit simulates the behavior of the Landing Vision System on the ground-to-camera transformation: in particular, a single landmark is identified by 5 points on its rim, for a total of up to 100 landmarks detected in the camera field of view. This arbitrary value takes into account a sufficient number of landmarks, considering possible missed recognitions. These extracted points undergo a pinhole transformation that generates ellipses shapes in the camera frame.

- **Camera-to-Ground Transformation**
  This block, in contrast to the previous one, performs the inverse transformation, extrapolating centers of the ellipses and their equivalent radius to trace back the shapes to their circular form. In this phase estimated altitude and attitude provide an additional reconstruction error that afflicts the landmark output list.

In summary, the simulation model outputs a set of up to 100 detected landmarks, each defined by its center coordinates and radius, serving as input for the subsequent matching algorithm.

The initial orientation of the vehicle is represented using a quaternion, a 4-dimensional vector that provides a compact and non-singular representation of rotation in 3-dimensional space. The quaternion is defined as follows:

$$\mathbf{q} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) \\ u_x \sin\left(\frac{\theta}{2}\right) \\ u_y \sin\left(\frac{\theta}{2}\right) \\ u_z \sin\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{5.3}$$

Where:

- θ is the rotation angle,
- $\mathbf{u}$ is the rotation axis (unit vector),
- q_0 is the scalar part of the quaternion,
- q_1, q_2, q_3 form the vector part of the quaternion.

The position of the spacecraft is represented by a 3-dimensional vector, which defines the coordinates of the spacecraft in the 3-Dimensional space. The position vector is defined as follows:

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{5.4}$$

With:

- x as the longitude coordinate,
- y as the latitude coordinate,
- z as the altitude coordinate.

Below, the Simulink model is presented in Figure 5.2.

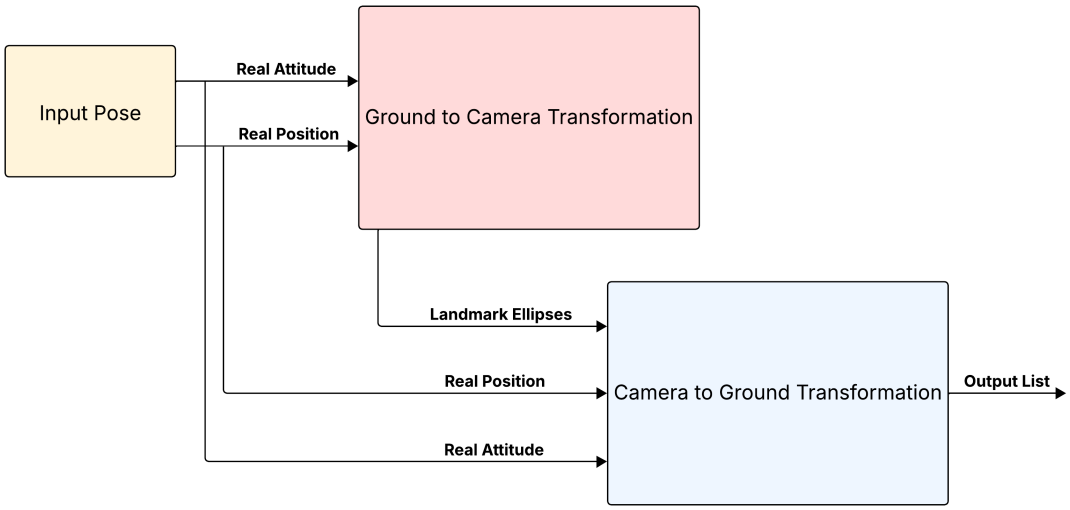

Figure 5.2: Top-level of the Simulink model

In particular, the Camera-to-Ground Transformation block elaborates the real pose of the spacecraft received from the input, generating the estimated attitude quaternion and the position vector adding uncertainties to real values, according to the error ranges established in advance.

To provide a clearer understanding of how the estimated initial position vector and estimated attitude quaternion are generated, the adopted approach is presented:

- The estimated initial position vector is defined as:

$$\mathbf{p}_{est} = \begin{bmatrix} x_{est} \\ y_{est} \\ z_{est} \end{bmatrix} \tag{5.5}$$

  Where:

  - $x_{est} = 0$
  - $y_{est} = 0$
  - $z_{est} = 4100$

  Latitude and longitude coordinates are set to zero as the spacecraft is assumed to be at the center of the image, placing accordingly its relative coordinate

system. Altitude is set at 4100, which is the detected elevation value at which the LVS begins its operations.

- The estimated attitude quaternion is defined as:

$$\mathbf{q}_{est} = \begin{bmatrix} q_{0,est} \\ q_{1,est} \\ q_{2,est} \\ q_{3,est} \end{bmatrix} \tag{5.6}$$

  Where:

  - $q_{1,est} = q_{1,act} + \Delta\phi$
  - $q_{2,est} = q_{2,act} + \Delta\theta$
  - $q_{3,est} = q_{3,act} + \Delta\psi$

Vectorial components of the estimated quaternion are composed of the real correspondent part with uncertainties added to each axis. The generation criteria for the uncertainties will be later analyzed in section 7.2.

Underneath, a closer view of Camera-to-Ground Transformation block is presented in Figure 5.3.

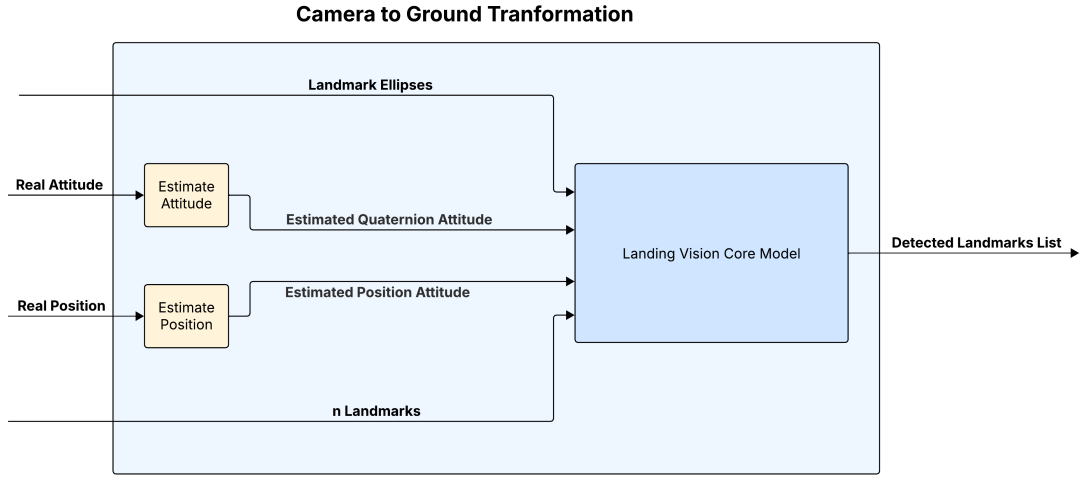

Figure 5.3: Camera-to-Ground Transformation block closer view

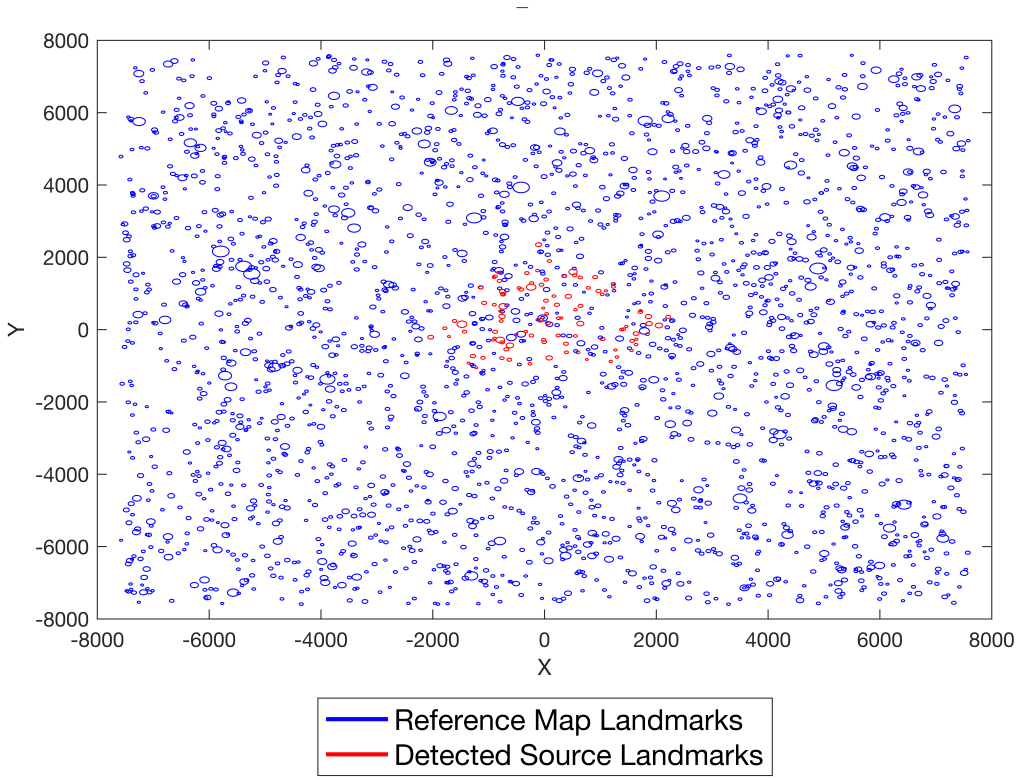

Figure 5.4: Example of detected landmarks in the reference map. An area of 4×4 km^2 is reported, centered on the estimated initial lander position and containing up to 100 detected landmarks.

Chapter 6

Matching Algorithms

This chapter provides an in-depth analysis of the matching phase previously outlined, presenting a detailed examination of two distinct methods. Both algorithms draw inspiration from existing literature, specifically Triangle Similarity Matching by Uwano et al. [10] and Neighborhood Matching by Li et al. [11].

These methods have been reinterpreted and further refined, with core features adapted to better suit a significantly different scenario, such as the pinpoint landing on Martian surface.

The decision to develop the two algorithms stems from their profoundly diverse approach towards the shared objective. TSM algorithm sets its foundations on a highly robust methodology, leading to an extremely accurate result in exchange for an elevate computation time. On the other hand, RANSAC algorithm goes in the opposite direction, fostering a significant low execution time but losing in reliability.

These contrasting solutions offer valuable insights into the pinpoint landing problem, presenting different perspectives on how to approach the same challenge.

6.1 Triangle Similarity Matching (TSM)

As the name suggests, this algorithm operates by constructing constellations composed of landmarks in the form of triangles. These triangular formations are derived from input data captured by the camera as illustrated in Figure 5.4, and

are subsequently compared against a reference set of precomputed constellations stored in the Landing Vision System. By leveraging this structured approach, the algorithm aims to establish correspondences between observed and known landmarks, exploiting geometric features of the triangles.

6.1.1 TSM Methodology

Initially, every extracted landmark obtained from the camera shot image is associated with the two nearest ones to model a triangle shape. For each constellation, salient features are obtained successively:

- **Angles** of the three inner vertexes
- **Lenght** of the three sides
- **Coordinates** of the three landmarks forming the triangle

Once the triangle set is obtained, the algorithm performs an initial discrimination between different objects in the catalogs by comparing the sum of the two smallest interior angles (6.1). If the difference between these sums falls below a predefined threshold, the two triangles can be considered approximately affine in the initial assessment.

$$\sum_{i=1}^{2} |\theta_i - \theta'_i| < DIFF \tag{6.1}$$

Where:

- θ_i is the i smallest inner angle of the known triangle
- θ'_i is the i smallest inner angle of the camera triangle
- DIFF is the given threshold value

Triangles identified as similar undergo further analysis to determine whether the match is accurate or a false positive. For each triangle, the algorithm computes three key features for the entire process:

- $\boldsymbol{c}$: geometric center of the triangle
- $\mathbf{d}_1$: vector of the longest side of the triangle

- $\boldsymbol{d}_{center}$: vector connecting the geometric center to the nearest landmark outside the triangle

These obtained values are then used to calculate the projection discrepancies in Equation 6.2 and Equation 6.3:

$$I = \left| \boldsymbol{d}_1 \cdot \boldsymbol{d}_{center} - \frac{\boldsymbol{d}'_1 \cdot \boldsymbol{d}'_{center}}{\gamma^2} \right| \tag{6.2}$$

$$C = \left| \boldsymbol{d}_1 \times \boldsymbol{d}_{center} - \frac{\boldsymbol{d}'_1 \times \boldsymbol{d}'_{center}}{\gamma^2} \right| \tag{6.3}$$

$$\gamma = \left| \frac{\boldsymbol{d}_1}{\boldsymbol{d}'_1} \right| \tag{6.4}$$

Where:

- I: the difference of the inner products between $\boldsymbol{d}_1$ and $\boldsymbol{d}_{center}$ of paired triangles

- C: the difference of the cross products between $\boldsymbol{d}_1$ and $\boldsymbol{d}_{center}$ of paired triangles

- γ: the scaling value between the pair

In this dissertation, γ is always considered ≈ 1 as the elevation uncertainty does not determine a consistent scaling factor between the pair.

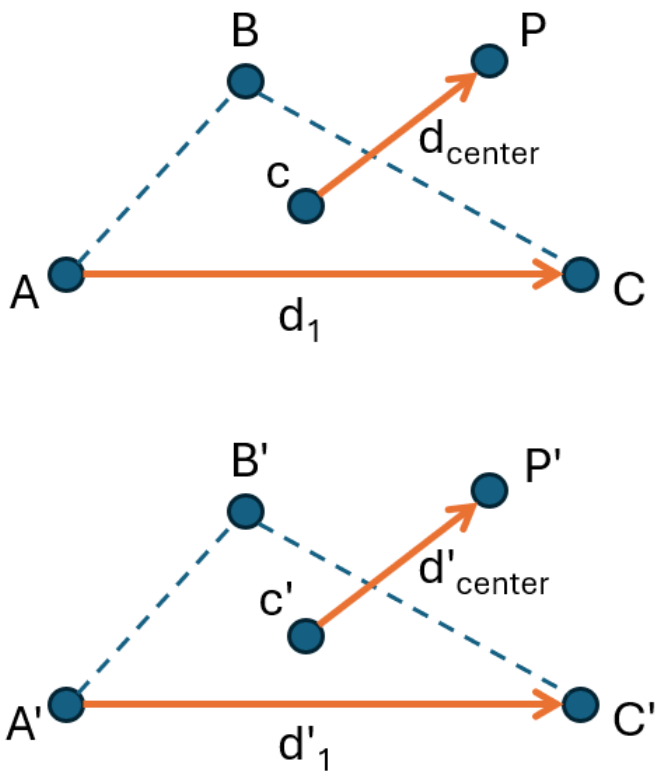

Figure 6.1: Triangle features

The key features extracted from the triangles are utilized to evaluate their similarity. Based on the reference literature study, a pair of triangles is considered matched if the condition specified in Equation 6.5 is satisfied [10]. This inequality serves as a critical criterion for determining the correspondence between the geometric properties of the triangles, ensuring that the matching process is both accurate and reliable. The satisfaction of this constraint confirms that the triangles share sufficient geometric similarity.

$$I^2 + C^2 < T \cdot |\boldsymbol{d}_{center}| \tag{6.5}$$

Where T is a fixed threshold. A preliminary list of paired triangles is thus available, as an example case in Figure 6.2 is depicted below.

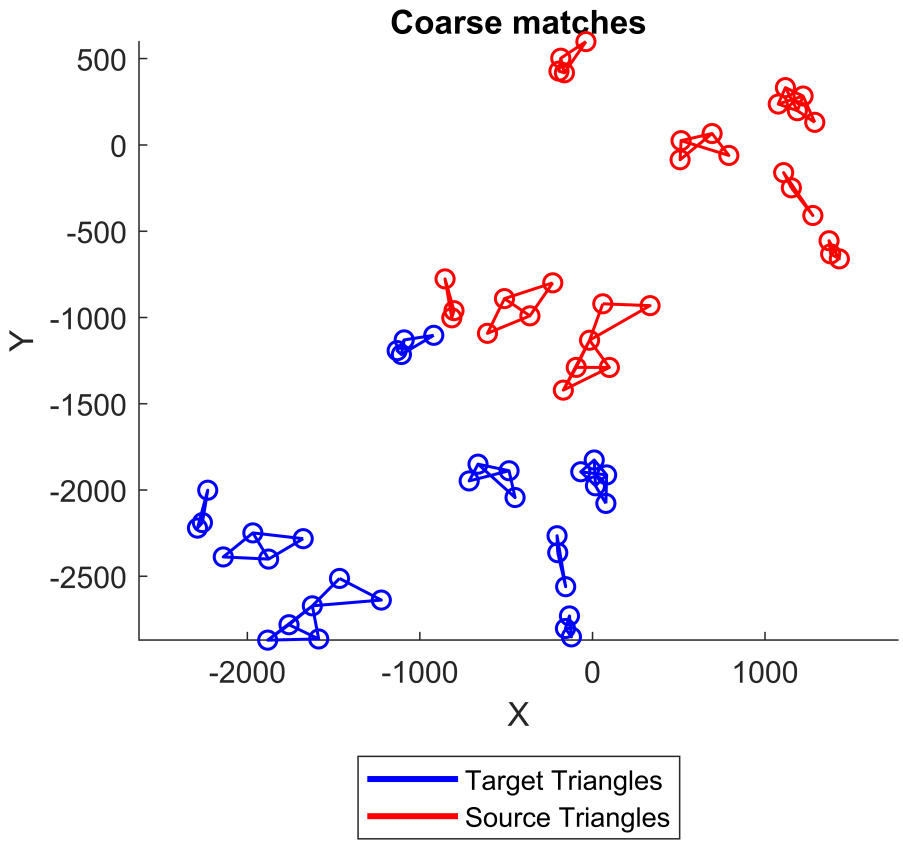

Figure 6.2: Example of preliminary TSM matched pairs

The final part of the algorithm involves a geometric transformation of the center point of the camera frame, which defines the vehicle position, to finally determine the estimated spacecraft coordinates with respect to the target point. To do so, the algorithm leverages the resulting similar triangle pairs from the aforementioned selection process, which are then used to compute the rotation matrix and the translation vector for the transformation, as described in Equation 6.6.

$$\boldsymbol{c}_t = \boldsymbol{R} \cdot \boldsymbol{c}_s + \boldsymbol{t} \tag{6.6}$$

Where:

- $\boldsymbol{c}_t$: target triangle geometric center in the preloaded map frame
- $\boldsymbol{c}_s$: source triangle geometric center in the camera frame

- $\boldsymbol{R}$: rotation matrix defined as follows:

$$\boldsymbol{R} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{6.7}$$

- θ: angle defining the rotation around an axis, perpendicular to terrain, joining the camera with the observed surface

- $\boldsymbol{t}$: translation vector

To compute the rotation matrix $\boldsymbol{R}$, a filtering strategy is adopted to extract the angle θ. This is achieved by calculating the angle between the geometric centers of all matched couples (6.8).

$$\theta_i = \arctan\left(\frac{y_{t_i} - y_{s_i}}{x_{t_i} - x_{s_i}}\right) \tag{6.8}$$

A normal distribution of these values is then formed, and the angles falling outside the 3–σ range are filtered out. The mean value of the remaining angles is subsequently used to define the rotation angle θ and thus the rotation matrix.

Successively, every triangle vertex and geometric center of the source frame is rotated by means of R to properly compute the translation vector. Distances between the rotated source centroids and the target ones are derived, and again a normal distribution of the results is generated. To ensure a robust estimation, the distances are filtered again excluding values beyond the 3–σ range. The mean value of the remaining distances is then used to define the translation vector $\boldsymbol{t}$. Below, Figure 6.3 shows how source triangles nearly overlap the target paired ones after the transformation following same example previously introduced.

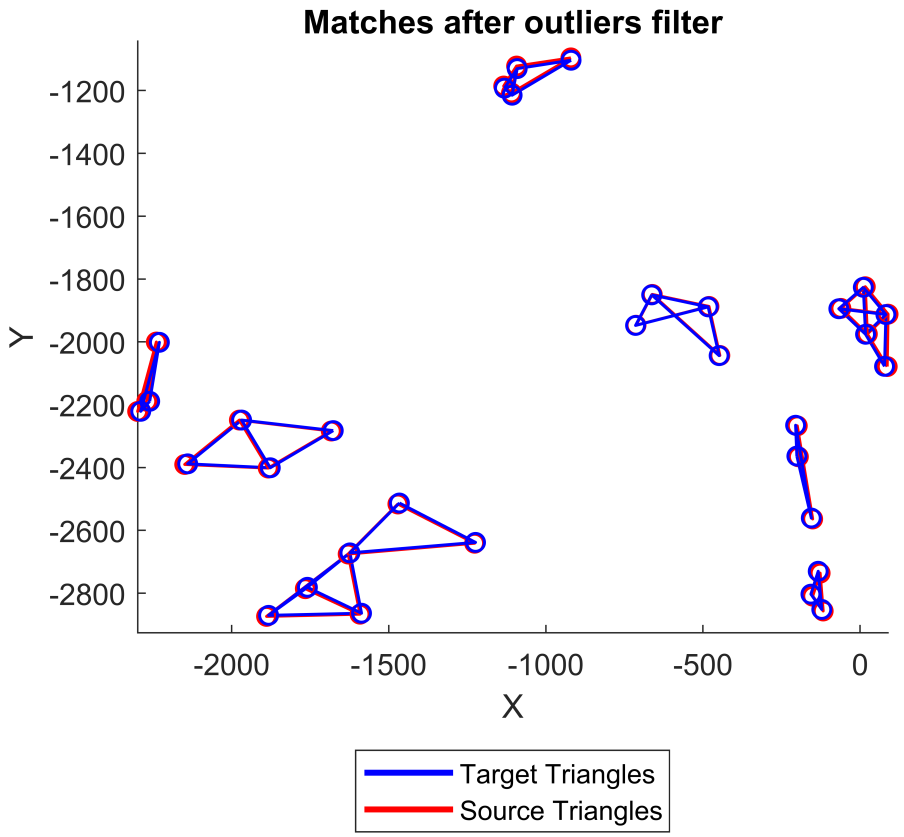

Figure 6.3: Example of refined TSM matched pairs

To ensure the reliability of the entire process, the algorithm performs an internal self-diagnosis to assess its current state. Specifically, it verifies whether the centroid distances, this time filtered using a more stringent 2–σ range, remain consistent with previously computed values. If the two distributions are found to be inconsistent, the algorithm flags the final result, signaling a potential invalid outcome in the matching process. The discrepancy is evaluated by means of a reliability index function, which takes into account the inliers ratio, the paired centroids displacement and the delta distance between the outputs derived from the two distributions, ensuring they fall under a predetermined threshold. A deeper analysis on this procedure in given in section 7.2.

This validation step is crucial for enhancing the practical application of the algorithm in pinpoint landing missions. Indeed, it introduces a layer of self-assessment within the process, helping to post-landing evaluations and troubleshooting, useful to the Mission Control Center to take decisions and elaborate recovery plans when

necessary.

Finally, the algorithm computes the estimated position of the spacecraft by applying the transformation defined in Equation 6.6 to the camera frame center point. This process yields the final coordinates of the spacecraft, providing the initial state that enables the trajectory planning and tracking to the target.

In Table 6.1 are summarized the parameters used in the algorithm, which are defined based on the analysis of the reference literature and the specific requirements of this application scenario. These values can be conveniently adjusted to optimize the performance of the algorithm for different situations, allowing for flexibility in the matching process.

Parameters	**Value**
DIFF	3
T	$4 \cdot 10^3$
γ	1

Table 6.1: Parameters used in TSM algorithm

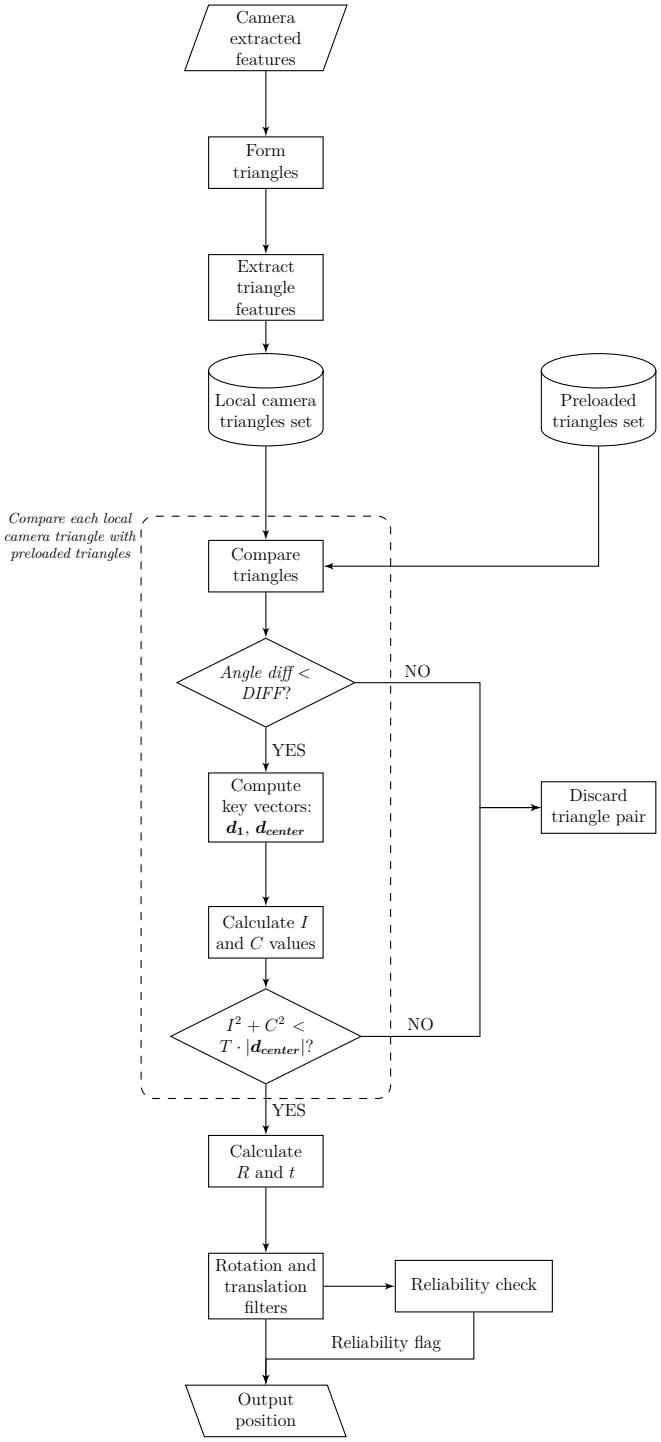

Figure 6.4: TSM algorithm flowchart

6.2 Neighborhood Matching (RANSAC)

This section presents a diverse matching algorithm, which differs significantly from the previous one in terms of approach and methodology. It is based on the best effort strategy, which aims to identify the best possible transformation that maps the camera frame planar projection to the correspondent one contained in the preloaded set of landmarks.

The above-mentioned algorithm, in fact, exploits the surroundings of each landmark to create a unique feature descriptor for each of them. These tuples, each composed of 3 features, create a 3-dimensional histogram that defines a neighborhood, which is then used to perform a preliminary list of similar landmark pairs. Successively, for each pair, a coarse matching and a refinement step approach are performed by means of Random Sample Consensus (RANSAC), leveraging the homography matrix to finally compute the best matching transformation.

6.2.1 RANSAC Methodology

To begin with, the algorithm identifies for each landmark in the camera frame a set of neighbors, which are defined as the landmarks falling within a certain distance from the selected one. This distance depends on the landmark radius, multiplied by a scaling factor k (6.9). In order to be considered valid, a neighborhood must contain at least m landmarks (6.10). Parameters k and m can be tuned opportunely to adapt the algorithm to different situations.

$$N_i = \{j \mid d_{i,j} < k \cdot r_i\} \tag{6.9}$$

$$|N_i| \geq m \tag{6.10}$$

Where:

- N_i: neighborhood of the i-th landmark
- $d_{i,j}$: distance between the i-th and j-th landmarks centers
- r_i: radius of the i-th landmark

- k: scaling factor
- m: minimum number of landmarks in the neighborhood

Once the neighborhoods are formed, key values are obtained for each set:

- The neighbor landmarks are sorted in descending order of size:

$$r_1 > r_2 > r_3 > \cdots > r_n$$

- $\mathbf{v}_1$: vector connecting the center of the landmark i to the center of the largest neighbor landmark j. This determines the local reference coordinate system, with the anticlockwise direction defined as positive.
- $\mathbf{v}_j$: vector connecting the center of the landmark i to the center of the n-th neighbor landmark.

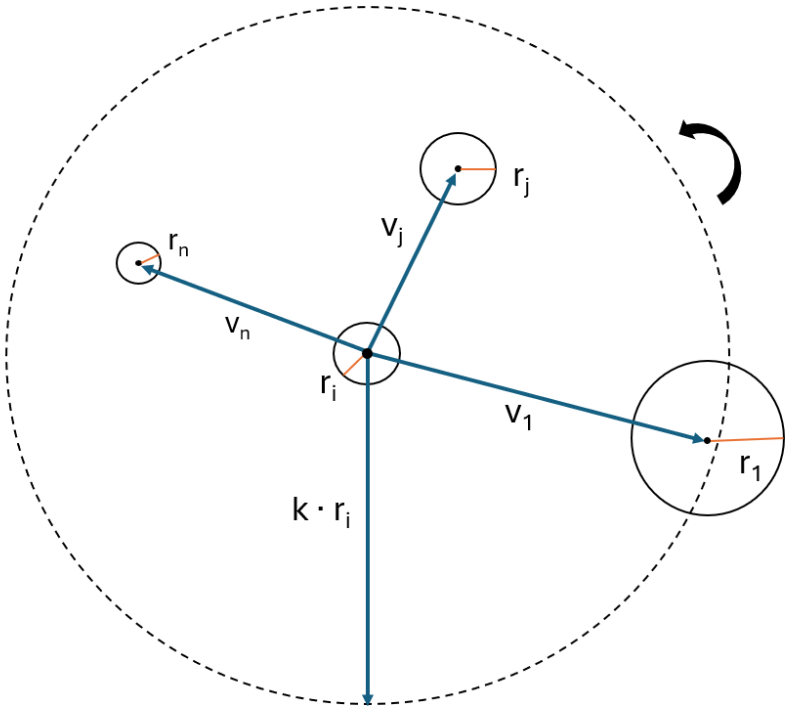

Figure 6.5: Example of neighborhood and its elements

Successively the algorithm computes three features for each neighborhood element, which are then used to generate a 3-dimensional histogram that uniquely describes the landmark i. The features are defined as follows:

1. **Orientation Feature:**

   - The angle θ_i between $\mathbf{v}_1$ and $\mathbf{v}_j$ is denoted as f_1^i, representing the orientation feature of j:

$$f_1^j = \angle(\mathbf{v}_1, \mathbf{v}_j)$$

2. **Scale Feature:**

   - The ratio of the center landmark radius to the radius of the neighbor j is computed as:

$$f_2^j = \frac{r_c}{r_j}$$

3. **Distance Feature:**

   - The ratio of the norm of $\mathbf{v}_j$ to the norm of $\mathbf{v}_1$, representing the relative distance feature of the neighbor landmark j, is defined as:

$$f_3^j = \frac{\|\mathbf{v}_j\|}{\|\mathbf{v}_1\|}$$

Tuples of features are thus obtained, each denoted as (f_1^j, f_2^j, f_3^j), and are further elaborated to create the histogram structures.

- Each feature f_1^j, f_2^j, and f_3^j is normalized to the range [0,1].
- The range [0,1] is divided into n_b discrete bins for each feature.
- For each neighbor landmark j, there are a total of n_b^3 possible combinations of feature values across the bins.
- The feature combinations of all neighbors are counted into a histogram H for landmark i.
- The histograms of all source landmarks form H_s, while the histograms of the target preloaded map form H_t.

The similarities between histograms in H_s and in H_t are measured by means of the Euclidean distance in the feature space. A correspondence relation is built if the Euclidean distance is less than a given threshold T.

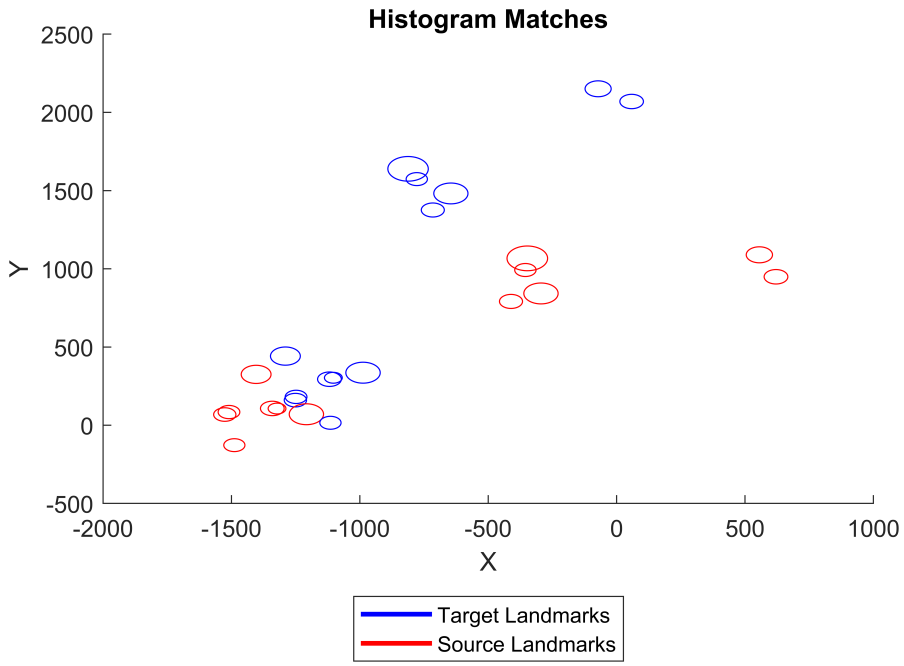

Figure 6.6: Example of initial matched pairs

With the paired landmarks list (source and target center coordinates), the algorithm proceeds to perform a coarse matching step. One pair at a time is selected and the Equation 6.11 is calculated:

$$\mathbf{c}_t = s \cdot \mathbf{R} \cdot \mathbf{c}_s + \mathbf{t} \tag{6.11}$$

Where:

- s is the scale ratio,
- $\mathbf{R}$ is the rotation matrix related to an angle φ,
- $\mathbf{t}$ is the translation vector.

The above equation defines a geometric transformation to perform a preliminary but approximate shift of the source landmarks towards the target ones. Let $\boldsymbol{c}_s$ be

the center coordinate of the source element, and $\boldsymbol{c}_t$ be its corresponding pair in the target reference frame. The scale ratio s can be determined as:

$$s = \frac{r_{c_s}}{r_{c_t}}$$

The angle φ can be determined by the angle between $\mathbf{v}_1^s$ and $\mathbf{v}_1^t$.

$$\varphi = \angle(\mathbf{v}_1^s, \mathbf{v}_1^t)$$

With s and φ, the translation vector $\mathbf{t}$ can also be determined. Every source landmark $\boldsymbol{c}_s$ is then transformed, generating a new set of coordinates C_s' in the target frame, where the single entry is defined as in Equation 6.12.

$$\boldsymbol{c}_s' = s \cdot \mathbf{R} \cdot \boldsymbol{c}_s + \mathbf{t} \tag{6.12}$$

After the coarse transformation, the source landmarks are roughly aligned with those in the target frame, leading to a more precise identification of correspondences. The matches between each element in the sets C_s' and C_t at this phase are determined based on the Euclidean distance between their centers.

To filter out mismatched correspondences, RANSAC method is exploited, which is utilized to reduce noise and outliers data, higlhy efficient to eliminate potential false matches. In the fine matching, the objective model of RANSAC is the transformation matrix between C_s' and C_t, which is a homography matrix as depicted in Equation 6.13.

$$\begin{bmatrix} x_t \\ y_t \\ 1 \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{bmatrix} \begin{bmatrix} x_s' \\ y_s' \\ 1 \end{bmatrix} \tag{6.13}$$

Where:

- (x_s', y_s') are the coordinates of the source landmarks after the coarse transformation
- (x_t, y_t) are the coordinates of the target landmarks
- $h_1, h_2, ..., h_8$ are the homography matrix coefficients

The algorithm at this point selects four pairs of landmarks, essential to solve the homography matrix. After obtaining the unknown values, all the correspondences of landmarks are used to compute the transformations of each coarse matched coordinate and a cost function is calculated (6.14).

$$\text{Cost} = \sum_{i=1}^{n} \left((x_{t_i} - x''_{s_i})^2 + (y_{t_i} - y''_{s_i})^2 \right) \tag{6.14}$$

$$x''_{s_i} = \frac{h_1 x'_{s_i} + h_2 y'_{s_i} + h_3}{h_7 x'_{s_i} + h_8 y'_{s_i} + 1}$$

$$y''_{s_i} = \frac{h_4 x'_{s_i} + h_5 y'_{s_i} + h_6}{h_7 x'_{s_i} + h_8 y'_{s_i} + 1}$$

Where:

- n is the total number of correspondences between C'_s and C_t
- (x_{t_i}, y_{t_i}) are the coordinates of the target landmarks
- (x''_{s_i}, y''_{s_i}) are the coordinates of the source landmarks after the fine transformation

In order to compensate the stochastic nature of this latest step and to increase the overall number of iterations, the algorithm is designed to repeat the random selection of four pairs and thus the computation of the homography matrix n_s times.

The procedures carried out from the coarse matching (6.11) up to the cost function calculation are iterated for each landmark pair in the preliminary list. The algorithm then selects the transformation that minimizes the cost function, and meaningful data is stored to obtain the final output:

- The best homography matrix
- The inlier pairs, which are the source correspondences $\boldsymbol{c}''_s$ (transformed by means of the selected best homography matrix) that produce a singular cost value below a defined threshold T_c
- The scaling factor s, rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{t}$ of the iteration that lead to the homography matrix with the lowest cost

The following Figure 6.7 presents an example of the best cost evolution over the iterations, highlighting the convergence of the algorithm toward a minimum value.

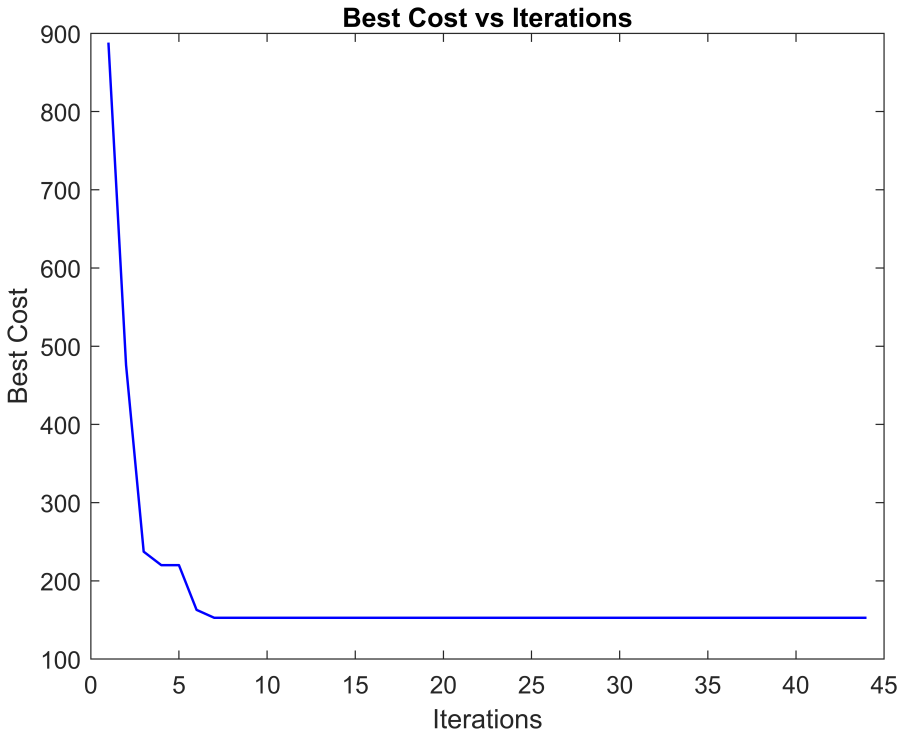

Figure 6.7: Best cost function trend over iterations

Before proceeding to produce the final output, similar to the TSM algorithm presented before, the process performs a self-diagnosis estimate about the reliability of the result. A linear correlation is inferred by experimental analysis between the overall Euclidean cost of the selected inliers and the distance that exists between the estimated position of the spacecraft and the target reference. An arbitrary reliability threshold T_r is thus defined, and the algorithm sets a flag reporting whether the matrix cost is higher or lower than the established threshold. The threshold is defined conservatively to ensure that a positive flag indicates a highly reliable result with a high degree of confidence, whereas a negative flag implies that the result is likely unreliable and should be interpreted with caution. Further insights are proposed in section 7.2. Eventually, both coarse and fine transformations with the selected parameters are applied to the camera frame origin point, leading to the final estimated position of the spacecraft.

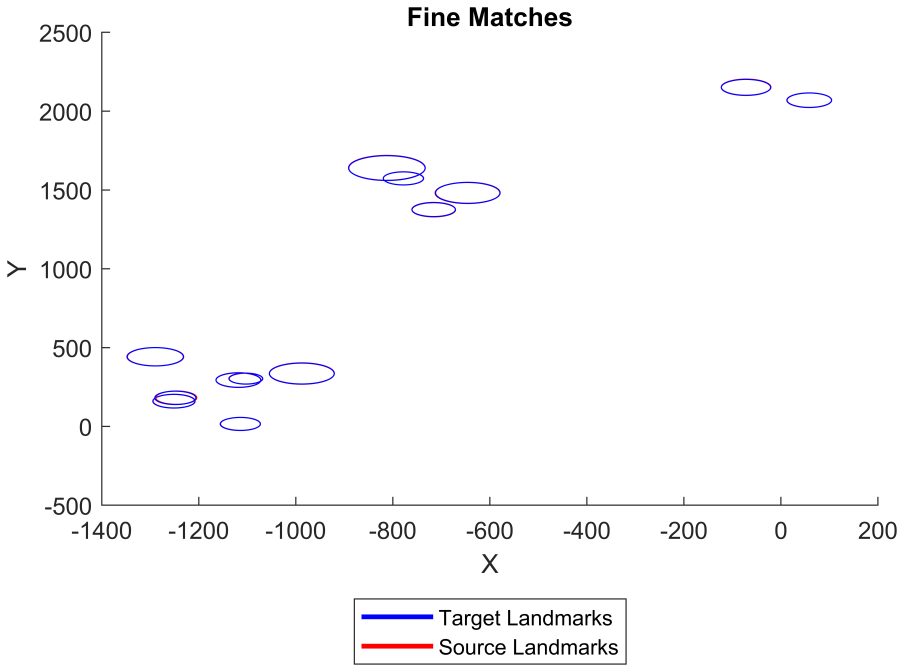

Figure 6.8: Example of final matched pairs

Below, Table 6.2 sums up the parameters utilized in the presented algorithm, which are tuned according to the best outcome of the experimental analysis for this specific application case.

Parameters	Value
k	7
m	5
n_b	4
T	0.7
T_c	$5 \cdot 10^4$
T_r	$1 \cdot 10^3$
n_s	4

Table 6.2: Parameters used in RANSAC algorithm

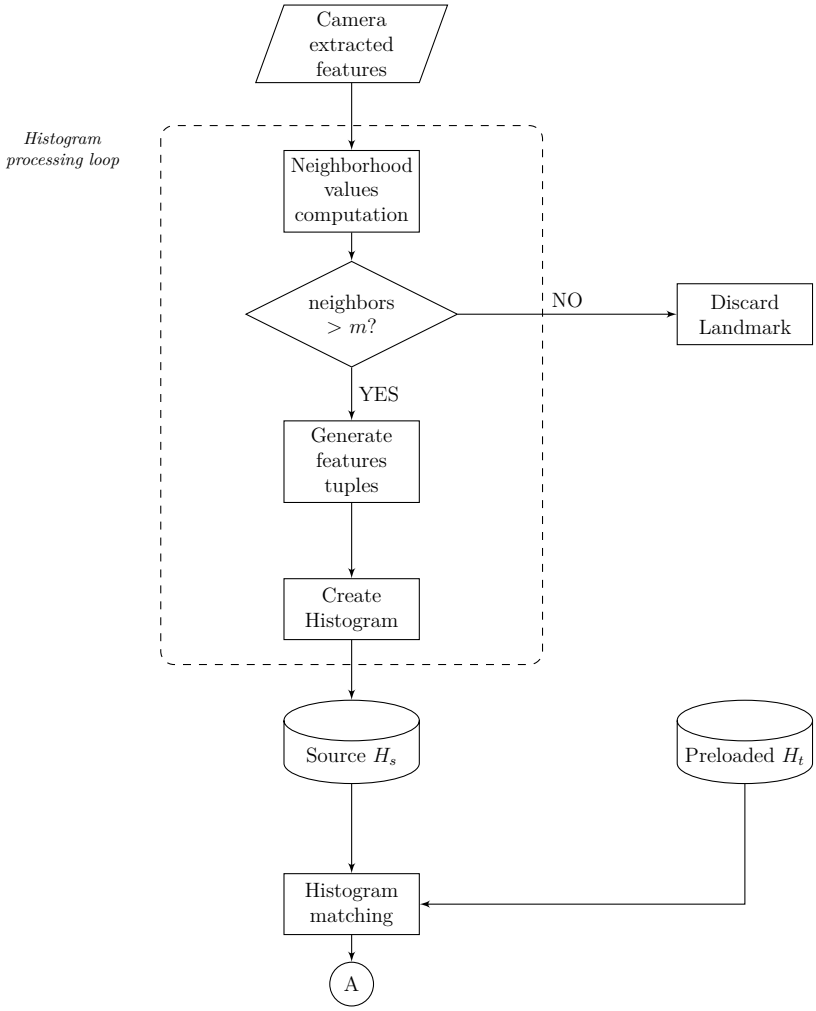

Figure 6.9: RANSAC algorithm flowchart - Preliminary Histogram Matching (Part 1)

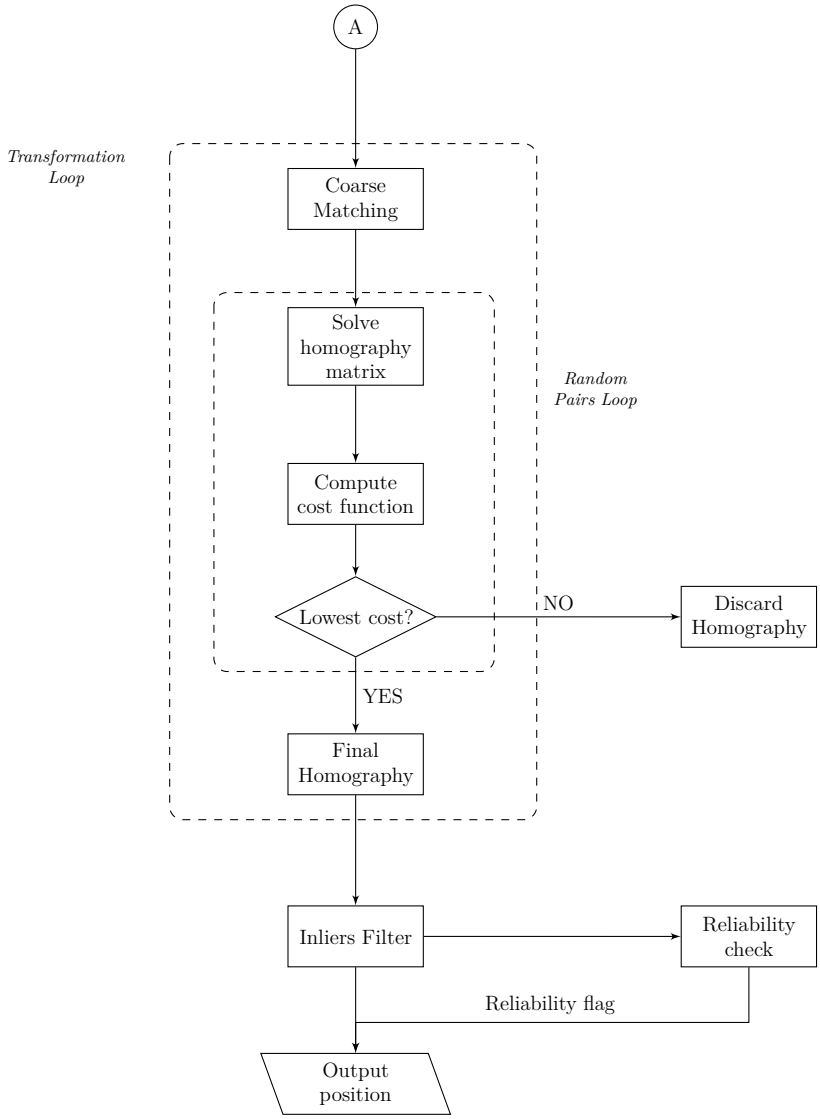

Figure 6.10: RANSAC algorithm flowchart - Coarse to fine matching output estimation (Part 2)

Chapter 7

Experimental Analysis

Meaningful results of the experimental verification conducted on both algorithms are presented in this section. A Monte Carlo analysis has been structured in order to evaluate the behavior in presence of uncertainties.

In order to provide a fair and coherent comparison between the two algorithms, the same tech stack and hardware architecture have been used for both. Here is a brief overview of the experimental setup:

- Programming Language and Environment:
  - MATLAB R2024b
  - Simulink
- System Architecture:
  - CPU: Apple M2
  - Memory: 8 GB SDRAM
  - Operating System: macOS Sequoia 15.3.1
- Execution Details:
  - Execution Mode: Single process
  - Threading Model: Single-threaded execution

As briefly mentioned in previous chapters, the matching process begins after the activation of the Radar Doppler Altimeter (RDA), at roughly 4100 m of altitude

where the LVS has few decades of seconds to establish the position of the spacecraft. Thanks to the stabilizing influence of the parachute, the lander in this phase is affected by a limited off-vertical angle. Kinematic states of the lander below the parachute, with the respective ranges and statistic distribution, are considered to define plausible and representative scenarios of investigation.

7.1 Nominal Case Analysis

Herein this section, the nominal scenario is analyzed. The nominal case is defined as the scenario where the system is free from any perturbations on the state variables of the model. Both algorithms are thus evaluated in an ideal situation, where the vehicle is perfectly aligned with the target landing point, at the correct altitude and with no banking or rotation variations. Results of both approaches are reported in Table 7.1.

Metrics	**TSM**	**RANSAC**
x [m]	0	0
y [m]	0	0
h [m]	4100	4100
ϕ_X [deg]	0	0
ϕ_Y[deg]	0	0
θ_Z[deg]	0	0
Distance Error [m]	0.24	7.34
Execution Time [s]	17.22	0.19

Table 7.1: Nominal Case Statistics

The Triangle Similarity Matching algorithm exhibits a lower error in the estimated position compared to the RANSAC approach, with a distance error of 0.24 m against 7.34 m. Nevertheless, the RANSAC algorithm shows a significantly lower execution time, with an average of 0.19 seconds, compared to around 17.22 seconds of the previously mentioned method. These insights are useful as they provide a baseline behavior of the system using the two distinct approaches, addressing the precision of both and the computation time needed to provide a solution.

7.2 Monte Carlo Analysis

The **Monte Carlo Analysis** is a statistical method utilized to assess the performances of the system with the impact of uncertainties on state variables. Such analysis has been conducted on the proposed simulation model for the two algorithms, with a total of 100 instances, each with a stochastic set of initial conditions. To provide a comprehensive and coherent comparison between the two methods, the same set of initial conditions has been used for both approaches, that is for the same run n corresponds an equal set of initial variable states.

- **Position** variations are modeled as a uniform distribution, assuming a maximum error of ± 3000 m in the two horizontal coordinates (approximately corresponding to latitude and longitude) with respect to the target landing point. It is important to note that 3000 m is a typical figure for the uncertainty linked to the Parachute Deployment Point at the end of the Guided Entry phase.

$$x_{Actual} = 0 + 6000 \cdot (rand - 0.5) \quad [m] \tag{7.1}$$

$$y_{Actual} = 0 + 6000 \cdot (rand - 0.5) \quad [m] \tag{7.2}$$

- **Altitude** uncertainties are modeled as a uniform distribution with a ± 65 m error on the estimated altitude of 4100 m, corresponding to the height at which the landmark detection and matching process begin. This error is a conservative evaluation of the altitude estimation uncertainty linked to the RDA inherent errors.

$$h_{estimated} = 4100 \quad [m] \tag{7.3}$$

$$h_{actual} = h_{estimated} + 130 \cdot (rand - 0.5) \quad [m] \tag{7.4}$$

- **Off-vertical angles** perturbations along the actual latitude and longitude are generated as a normal distribution with a maximum variation of ± 5 degrees on both axes of the spacecraft at 3–σ, while the error of knowledge (estimated off-vertical angles w.r.t actual off-vertical angles) is 1 deg at 3–σ.

$$\phi_{X_{actual}} = 0 + 10 \cdot (randn - 0.5) \quad [deg] \tag{7.5}$$

$$\phi_{Y_{actual}} = 0 + 10 \cdot (randn - 0.5) \quad [deg] \tag{7.6}$$

$$\phi_{X_{estimated}} = 0 + 2 \cdot (randn - 0.5) \quad [deg] \tag{7.7}$$

$$\phi_{Y_{estimated}} = 0 + 2 \cdot (randn - 0.5) \quad [deg] \tag{7.8}$$

- **Perpendicular angle** error with respect to the surface is instead modeled as a uniform distribution providing at most a ± 180 degrees change for the actual rotation along Z-axis. For what concerns the estimated value, the same approach has been adopted, adding a maximum margin of error of ± 1 degrees.

$$\theta_{Z_{actual}} = 0 + 360 \cdot (rand - 0.5) \quad [deg] \tag{7.9}$$

$$\theta_{Z_{estimated}} = 0 + 2 \cdot (randn - 0.5) \quad [deg] \tag{7.10}$$

The aforementioned uncertainties are summarized in Table 7.2:

Parameters	**Nominal Value**	**Distribution**	**Max Error** / 3σ
x_{act} [m]	0	Uniform	± 3000
y_{act} [m]	0	Uniform	± 3000
h_{act} [m]	4100	Uniform	±65
$\phi_{X_{act}}$ [deg]	0	Normal	±5
$\phi_{Y_{act}}$[deg]	0	Normal	±5
$\theta_{Z_{act}}$[deg]	0	Uniform	±180
$\phi_{X_{est}}$[deg]	$\phi_{X_{act}}$	Normal	±1
$\phi_{Y_{est}}$[deg]	$\phi_{Y_{act}}$	Normal	±1
$\theta_{Z_{est}}$[deg]	$\theta_{Z_{act}}$	Normal	±1

Table 7.2: Monte Carlo Analysis Errors

7.2.1 Results

In this section results obtained from the Monte Carlo analysis are exhibited, according to conditions described in section 7.2. In Figure 7.1 and Figure 7.2 shown below, the success rate of the two algorithms is depicted, highlighting the percentage of successful, thus valid, outcome over the total number of instances. Attention is also given to the number of algorithm failures—that is, instances where the process fails to produce a solution—and not valid cases where the estimated output distance is excessively far from the target.

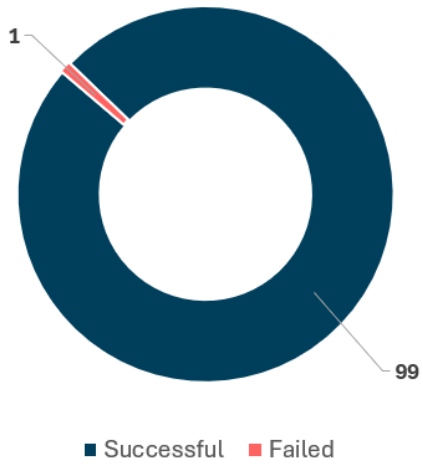

Figure 7.1: TSM Success Rate

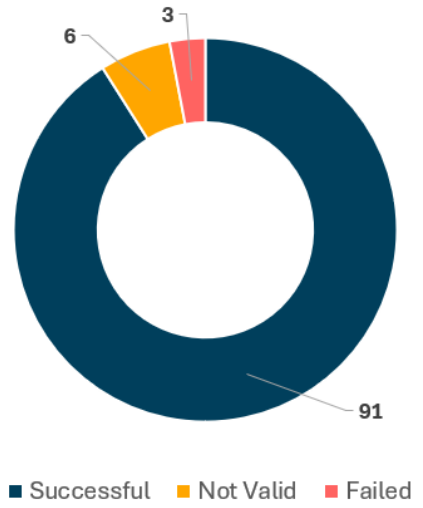

Figure 7.2: RANSAC Success Rate

Triangle Similarity Matching algorithm shows a higher success rate compared to the RANSAC approach, with a percentage of 99%, 1% failure and no invalid cases. On the other hand, the RANSAC algorithm exhibits a lower success rate, with a percentage of 91% valid cases, 3% of failures and 6% of invalid cases. In the following figures, only valid cases are taken into account for both algorithms, in order to provide a significant comparison of performances in positive outcomes, while false

positives are introduced when needed to supply more detailed insights. As shown in Figure 7.3 and Figure 7.4 the two methods exhibits similar performances in terms of accuracy of the estimated spacecraft position.

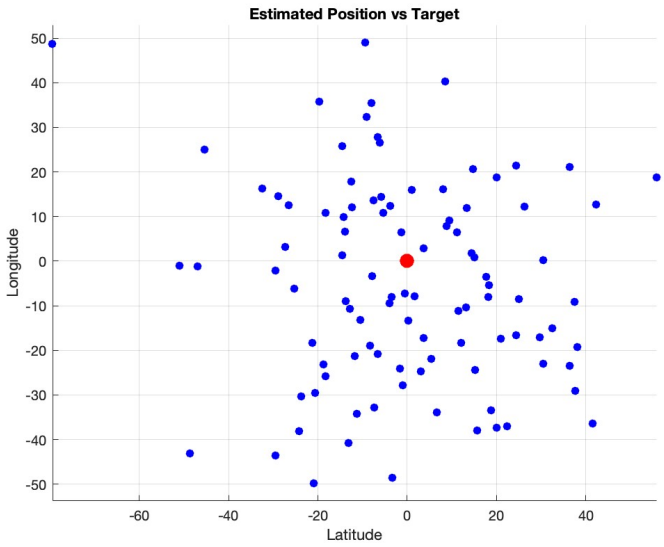

Figure 7.3: TSM Scatter Plot

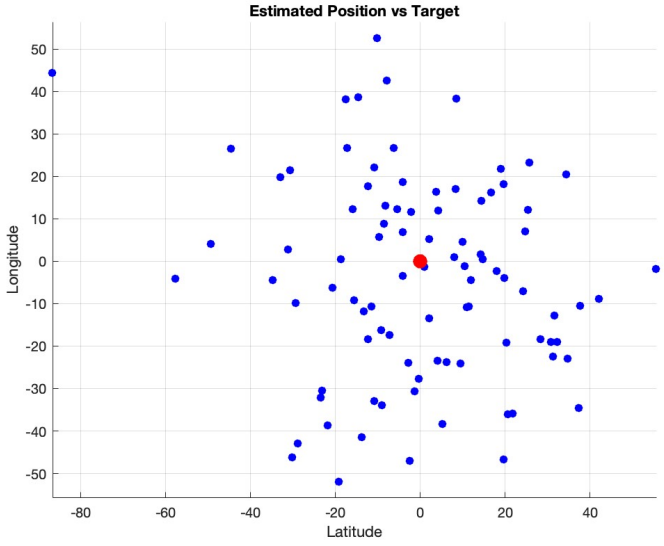

Figure 7.4: RANSAC Scatter Plot

Herein below, in Figure 7.5 and in Figure 7.6, histograms of the estimated position error with respect to the target are shown for both algorithms.

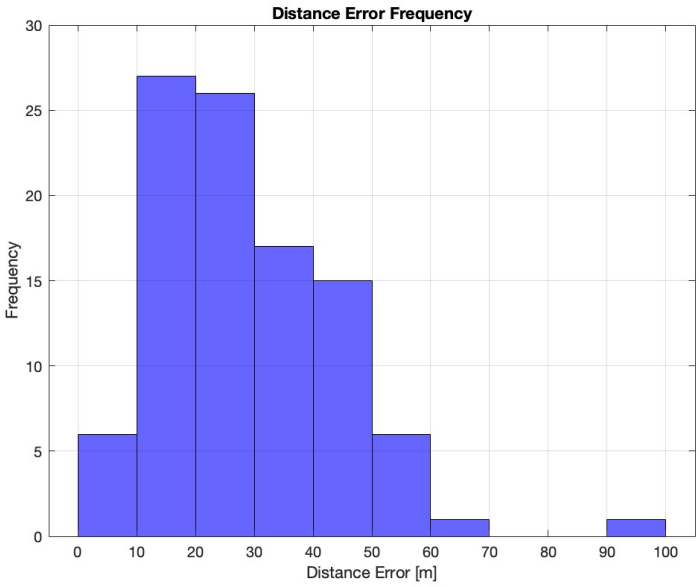

Figure 7.5: TSM Estimated Position Error

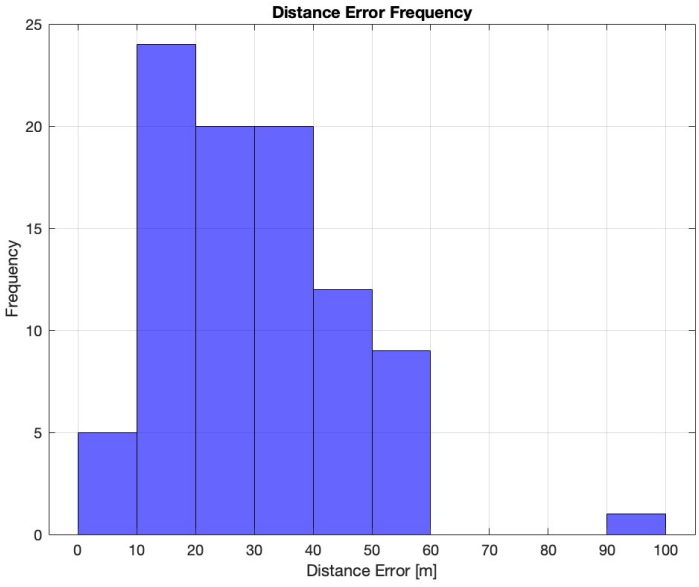

Figure 7.6: RANSAC Estimated Position Error

If only valid cases are considered, similar behaviors are observed using the two distinct approaches. Although the RANSAC algorithm demonstrates a slightly higher error tendency in the estimated position, if the invalid cases are taken into account, as Figure 7.7 depicts, nevertheless maintaining a high accuracy overall.

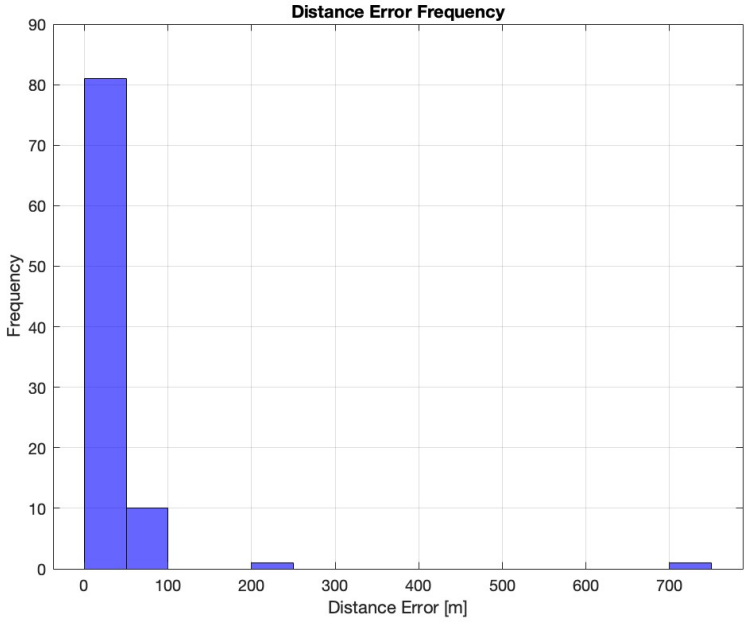

Figure 7.7: RANSAC Estimated Position Error (with outliers)

Figure 7.8 and Figure 7.9 highlight the similarity in the percentile distribution of the estimated position error for the two methods in the valid cases, keeping the 97% of distance gaps below 60 m.

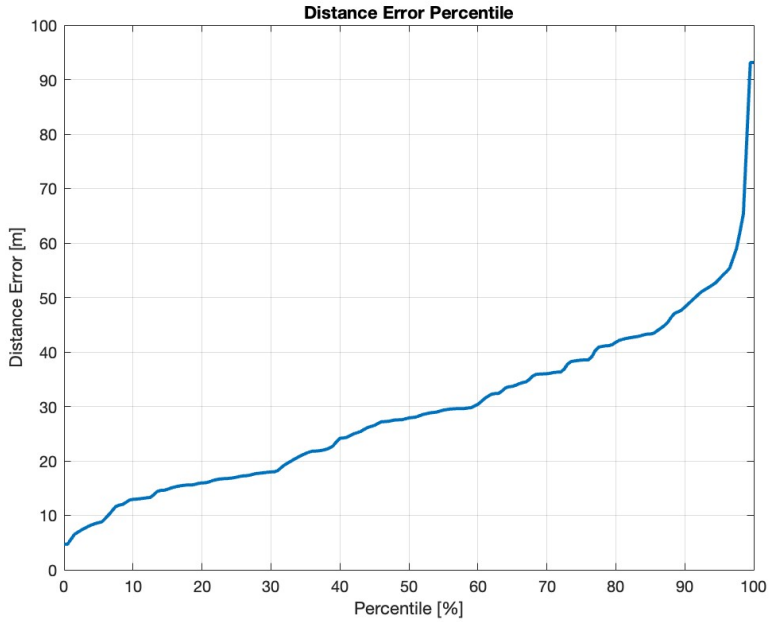

Figure 7.8: TSM Estimated Position Error Percentile

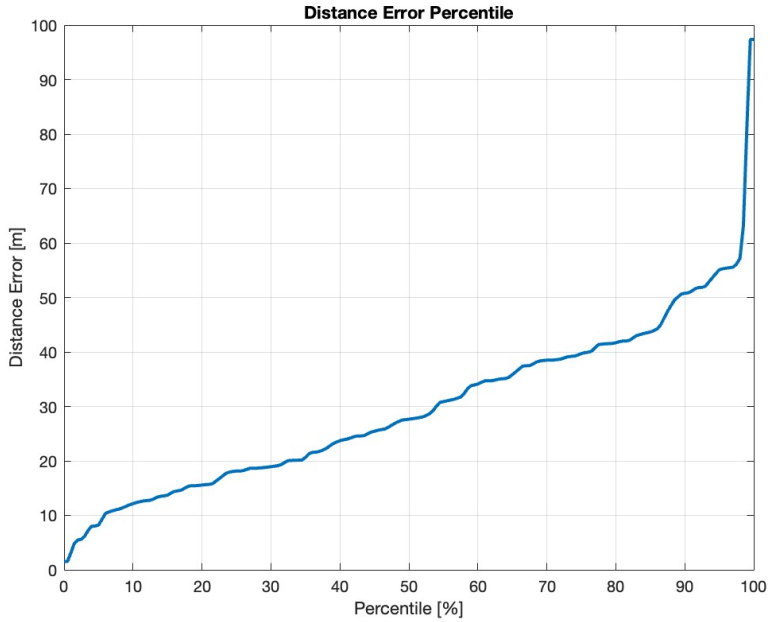

Figure 7.9: RANSAC Estimated Position Error Percentile

A box-plot is useful to provide a clearer understanding of the estimated distance error distribution, as shown in Figure 7.10 and Figure 7.11 for valid cases, and in Figure 7.12 encompassing also invalid cases.

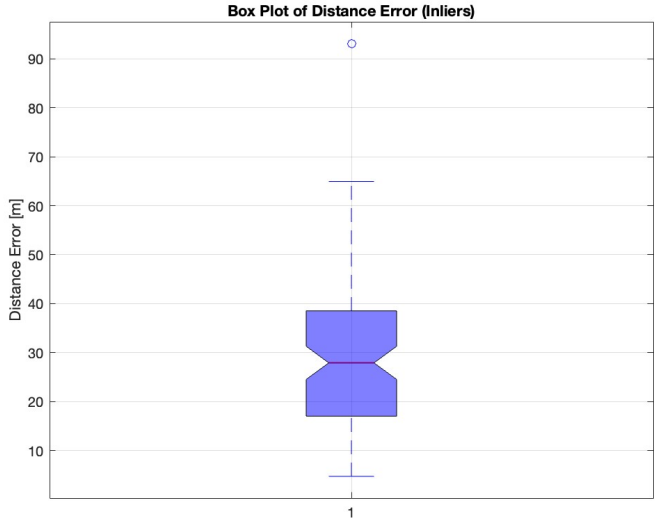

Figure 7.10: TSM Estimated Position Error Box-Plot

Figure 7.11: RANSAC Estimated Position Error Box-Plot

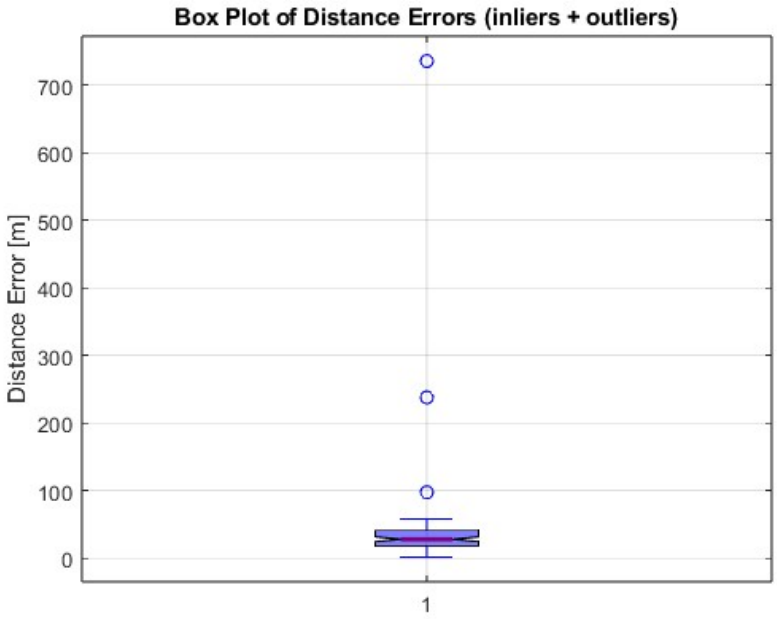

Figure 7.12: RANSAC Estimated Position Error Box-Plot (with outliers)

In the following images are visually reported the criteria used to select reliable outcomes for both algorithms. For the Neighborhood algorithm results has shown that a reasonable conservative threshold for the final inliers cost is $T_r = 1 \cdot 10^3$, as it guarantees an acceptable trade-off between the number of false positive outcomes avoided and the number of valid cases discharged (Figure 7.13 and Figure 7.14).

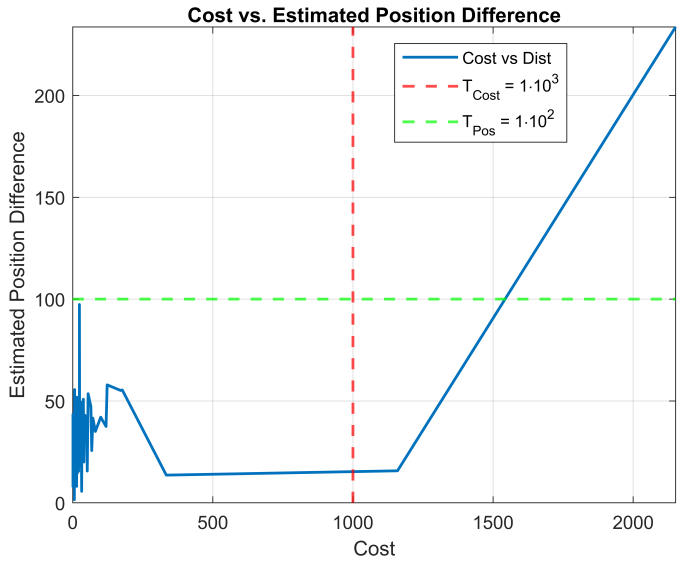

Figure 7.13: RANSAC Reliable Cases close-up view

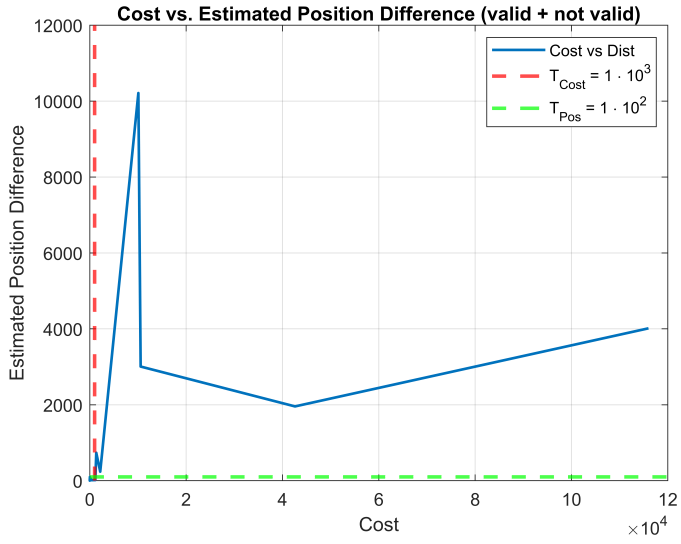

Figure 7.14: RANSAC Reliable Cases complete view

On the other hand, for the TSM algorithm a reliability index is computed at the end of the matching process, taking into account significant values obtained. The reliability index is so calculated:

$$I_{rel} = 0.5 \cdot I_{ratio} + 0.25 \cdot S_{dist} + 0.25 \cdot \Delta_{origin} \tag{7.11}$$

Where:

- I_{ratio} is the ratio between the number of final inliers and the number of initially matched pairs

- S_{dist} is the mean distance between centroids of the 2–σ and 3–σ filtered pairs

- Δ_{origin} is the difference between the transformed origin point with 2–σ and 3–σ filtered sets mean distance, respectively

The analysis of the results demonstrates that the proposed reliability index effectively reflects the validity of the algorithm final outcome. In this specific scenario, a reliability threshold of $T_{rel} = 3$ can be reasonably defined as an upper limit to consider the result valid.

Below, in Figure 7.15, every valid case is depicted by means of its reliability index and distance error, drawing attention to the arbitrary reliable area and the estimated distances falling within it.

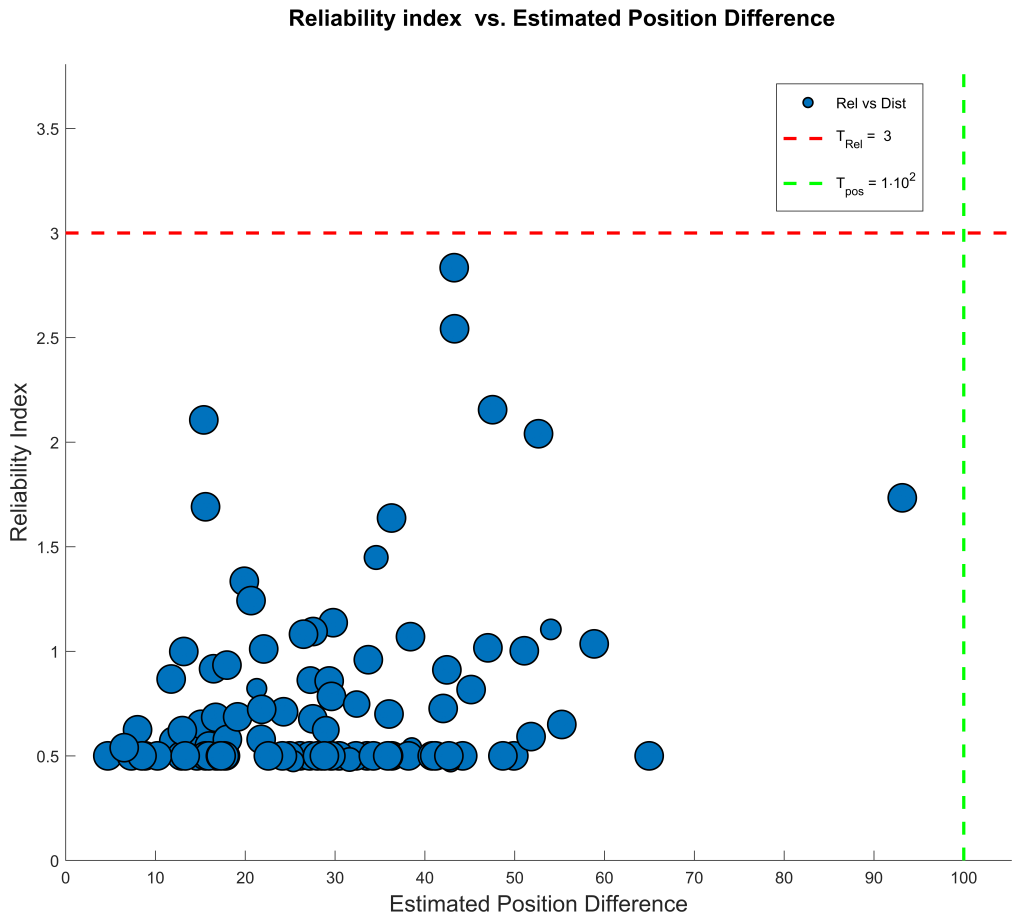

Figure 7.15: TSM Reliability Index vs Estimated Position Difference

Finally, in Figure 7.16 and Figure 7.17 presented herein below, a crucial result for this study is pointed out. Although the TSM algorithm exhibits a higher success rate, along with a lower error in the estimated position, the RANSAC approach shows a significantly lower execution time, with an average of roughly 0.2 seconds, compared to around 15 seconds of the previously mentioned method. This result legitimates the validity of the RANSAC algorithm, despite lacking of extreme accuracy, as it is capable of providing a still reliable solution in a substantially shorter time.

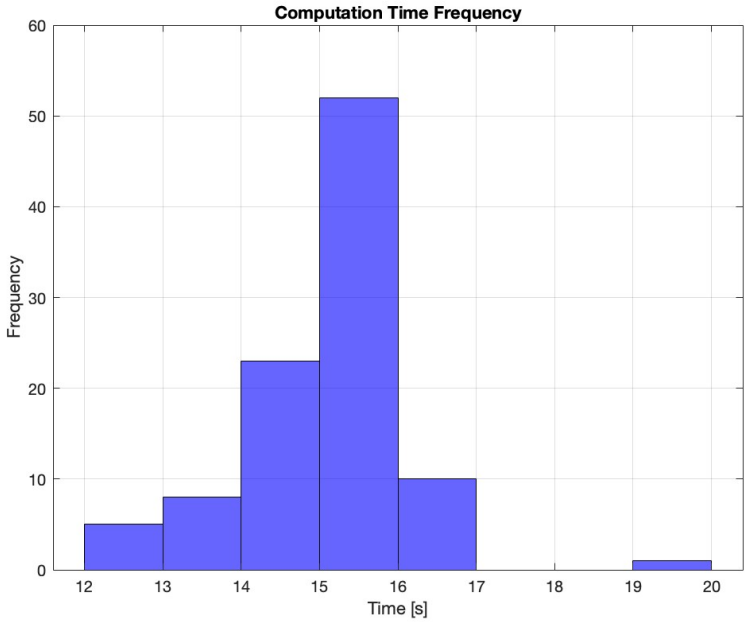

Figure 7.16: TSM Execution Time

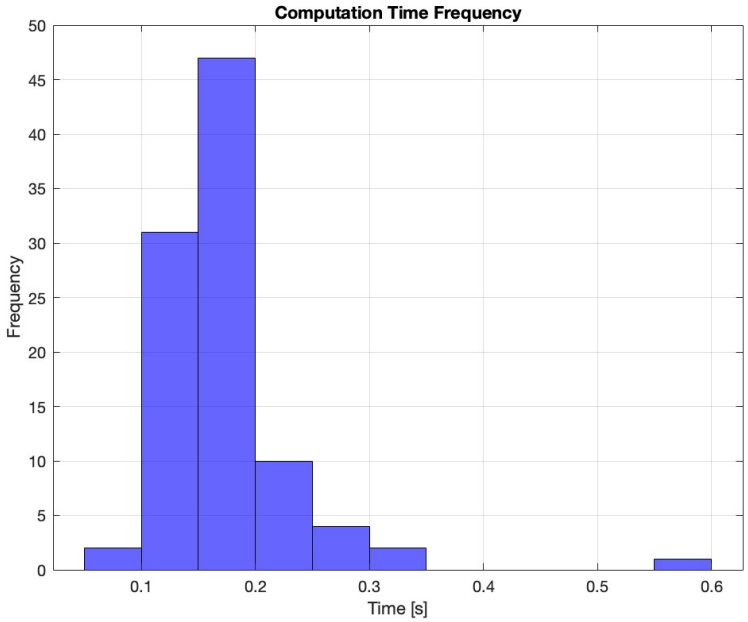

Figure 7.17: RANSAC Execution Time

The Table 7.3 summarizes all the results achieved by both algorithms. While the TSM method is highly robust and accurate in providing a reliable and satisfactory solution, it is also computationally demanding and time-consuming. Indeed, the dramatic advantage of RANSAC in terms of computation efficiency suggests the implementation of an iterative approach, in case of failure, processing successive image of the terrain. For instance the hypothesis of 2 successive elaborations, with an allocated time ranging to a maximum of 1.2 s (0.6 s $\cdot$ 2), the probability to fail the localization in both attempts would be 0.09^2, i.e. 0.0081. Therefore, the RANSAC would be capable to guarantee a success probability of 99.2% in only 1.2 s against the TSM able to provide an equivalent success probability of 99% in about 15 s. Furthermore, it could be soon evident that a wide margin would be present in the RANSAC case for allocating successive iterations, thus with the possibility to make negligible the failure probability. Instead, with the TSM algorithm, also a single iteration after the first execution would not be allowed due to the time constraint of the landing phase, since the lander would be already flying at an altitude of about 2400 m where the target should be already identified, the trajectory planned, and the controlled phase started.

Metrics	**TSM**	**RANSAC**
Mean Distance Error [m]	29.16	29.66
Median Distance Error [m]	27.89	27.65
Max Distance Error [m]	93.152	97.36
Min Distance Error [m]	4.7	1.55
Mean Execution Time [s]	15.16	0.17

Table 7.3: Algorithms statistics

7.3 Future Developments

Further refinements and enhancements can be applied to both the Triangle Similarity Matching (TSM) and RANSAC algorithms to enhance their performance and usability. The primary objective is to mitigate the computational drawbacks of the TSM algorithm while simultaneously improving the robustness and accuracy of the RANSAC approach. By addressing these aspects, the overall effectiveness of the target localization strategy can be significantly improved.

7.3.1 Enhancements of the TSM algorithm

TSM algorithm results in a demanding computation time due to the large number of combinations it has to perform to generate triangles: in fact, for every element a full search among the list of landmarks is required to find the two closest ones. Moreover, a consistent number of feature is calculated by means of trigonometric functions, which add up a not so negligible cost of time to the whole process. Eventually, the matching process between generated triangles entails a massive number of iterations to find the best fitting pairs, with a worst case scenario of $O(n^2)$ complexity.

To address these issues, several improvements can be employed:

- **Clustering-Based Triangle Filtering**: By grouping triangles with similar geometric properties into clusters, the number of candidate triangles for matching can be significantly reduced. This approach narrows down the search space and expedites the comparison process.
- **Kd-Tree Data Structure**: Implementing a kd-tree for spatial indexing of landmarks can greatly enhance the efficiency of nearest-neighbor searches. This structure enables logarithmic search time, reducing the overall computational burden and making the TSM algorithm more feasible for time-sensitive applications.
- **Parallel Processing and Vectorization**: Exploiting modern computing capabilities, such as multi-threading and GPU acceleration, can distribute the computational workload efficiently if the on-board instrumentation is equipped with dedicated hardware. This would allow for concurrent triangle formation and feature extraction, reducing execution time

By incorporating these techniques, the TSM algorithm can achieve a more competitive execution time, making it suitable for a larger set of mission scenarios.

7.3.2 Enhancements of the RANSAC algorithm

The RANSAC algorithm, while effective in handling outliers matched pairs, is inherently less accurate than the TSM approach due to its probabilistic nature. To improve its reliability and ensure consistency in landmark matching, an advanced hierarchical structure can be introduced. This framework introduces a multilayer iterative approach that handles the Neighborhood engine to multiply the opportunities of matching computation and process and endorses the final outcome of the algorithm.

The proposed framework operates as follows:

- **Hierarchical Supervision**: The algorithm is structured into at least two levels:
  - The lower level consists of the core RANSAC implementation, responsible for the estimated position output. At this level, a metrics, based on a functional cost linked to the inner algorithmic computations, allows the generation of a boolean indicator. The threshold is fixed in such manner that the outliers may be detected and discharged. Due to the fact that the limit could in some cases declare as not reliable an acceptable result, the hereafter described mechanization is helpful to complete the procedure successfully.
  - The upper level function handles the boolean flag together with possible iterations and a majority voting criterion to achieve the final algorithmic result.
- **Iterative Reliability Assessment**:
  - The RANSAC algorithm is executed up to three times.
  - After each iteration, the boolean reliability indicator provided by the core method is checked.
  - If the result is deemed reliable, the estimated position is accepted immediately.

  - If not, another iteration is performed, with a maximum of three attempts.

- **Fallback and Majority Voting Mechanism**:
  - If after two iterations no reliable solution is found, the third iteration is accepted as the final output if deemed valid.
  - If all three iterations yield unreliable results, a majority voting mechanism is employed. The most different estimation among the three ones is discharged and the most recent among the remaining two ones is chosen as final result. It is considered as the one better reflecting the spacecraft state.

By integrating this hierarchical approach, the RANSAC algorithm gains an additional layer of robustness. The supervision ensures that unreliable results are filtered out, increasing overall accuracy without compromising the efficiency of the method.

Indeed, the image processed at each iteration is different from the ones processed in the previous algorithm steps, therefore the probability of error can be considered as the combined probability of the error in each successive iteration determining a huge reduction of failed matching events after three iterations.

In conclusion, according to these presented enhancements, both the TSM and RANSAC algorithms can be significantly improved in terms of computational efficiency and robustness. The TSM algorithm, with the aid of kd-trees and clustering techniques, can achieve faster triangle matching. Meanwhile, the supervised RANSAC framework increases the reliability of landmark matching by incorporating a structured decision-making process. Together, these improvements will contribute to a more efficient and accurate initial localization strategy, enabling pinpoint landing with fine precision in the future planetary missions.

Chapter 8

Conclusion

This dissertation is dedicated to the development and evaluation of a simulated scenario for a pinpoint landing mission within a Martian environment. Particular emphasis is placed on the design, implementation, and analysis of the matching algorithm process, which plays a pivotal role in determining the pose of the vehicle relative to a predefined reference frame. This process enables the constrained optimized trajectory planning and the terrain-relative navigation system to accurately guide the spacecraft toward the designated landing site with high precision.

In pursuit of this objective, two distinct algorithmic approaches have been explored, each developed to address the challenges associated with terrain-relative navigation in different operational conditions. These methodologies aim to offer complementary advantages, ensuring a robust and adaptable solution for a variety of mission scenarios. By leveraging the strengths of both approaches within an integrated framework, a more reliable and efficient solution can be achieved, enhancing the overall performance of the landing system.

The first approach, the Triangle Similarity Matching (TSM) algorithm, has demonstrated a high degree of robustness and accuracy in estimating the position of the spacecraft. As evidenced by the results obtained from Monte Carlo simulations, discussed in Section 7.2, this method exhibits a strong capability to reliably determine the vehicle pose with a high level of certainty. However, despite its accuracy and robustness, the computational expense associated with the TSM algorithm poses a significant challenge. The processing time appears to be a critical limiting factor. The observed computation time approaches the maximum duration

allocated for target localization in a pinpoint landing mission. Consequently, while the current performance would, in principle, allow for the execution of a successful pinpoint landing, the possibility of performing a second iteration in case of an uncertain outcome would not be feasible. Therefore, improvements should primarily focus on enhancing computational efficiency, with the objective of completing the processing within 7 to 8 seconds.
It is remembered, in this passage, that the computation can be initiated only when the front-shield release allows the activation of the altimeter and the obtaining of the surface images. When these operations are completed the altitude of the lander is slightly over 4000 m. On the other hand the target elaboration must be completed when the lander is at about 2500 m of altitude to allow the orientation of the spacecraft and the initiation of the divert manoeuvre at about 2000 m of altitude. Observing that the trajectory planning can take also some minimum time for its elaboration, with a descent velocity of 75 m/s under parachute, the maximum time at disposal of the localization algorithms is in the order of about 15 s.

Conversely, the second approach, based on the Random Sample Consensus (RANSAC) algorithm, offers a more computationally efficient solution. This method provides a reliable estimation of the spacecraft position at a significantly reduced computational cost—approximately 100 times faster than the TSM algorithm. Such efficiency makes it a viable option for real-time applications and scenarios with stringent time constraints. However, while the RANSAC algorithm delivers acceptable accuracy, it does not achieve the same level of precision as the TSM algorithm. This reduced accuracy could result in minor deviations from the optimal landing trajectory, potentially impacting the final landing precision.

Despite these trade-offs, the domain of pinpoint landing on Mars remains an area of ongoing research and rapid evolution. With continuous advancements in optimization techniques and artificial intelligence-driven strategies, new methodologies are emerging to enhance landing accuracy and efficiency. The two algorithms presented in this study establish a solid foundation for further research in this domain. Their complementary nature suggests that a hybrid approach—leveraging the accuracy of TSM in scenarios where time is not a critical constraint and the speed of RANSAC in real-time applications—could lead to a more balanced and effective solution.

By contributing to the relatively unexplored yet crucial field of Martian pinpoint landing, this research provides a concrete basis for future studies. The findings and methodologies outlined herein serve as a stepping stone for continued advancements in planetary navigation, ensuring safer and more precise landings on extraterrestrial surfaces in future missions.

Appendix A

Appendix - TSM Algorithm Pseudo-code

Algorithm 1 Triangle Similarity Matching (TSM) Pseudo-code

I. Data Acquisition and Preprocessing
This phase retrieves and processes image data creating triangles. Structures and thresholds for the matching are initialized.

A. Data Retrieval
1: Retrieve triangle map from reference catalog
2: Obtain landmark map with feature measurements from source image
B. Image Processing
3: Generate triangles with geometric features from source landmark catalog
C. Parameter Initialization
4: Configure similarity thresholds: $DIFF \leftarrow 4 \times 10^3$, $T \leftarrow 3$, $\gamma \leftarrow 1$
5: Initialize parameters and data structures

II. Triangle Correspondence Identification

This phase identifies potential triangle matches based on angular similarity and geometric features.

A. Angular Similarity Assessment

for each camera-derived triangle T_i^s **do**
    Extract the two smallest interior angles $\alpha_{shot} \leftarrow \{\alpha_1, \alpha_2\}$ where $\alpha_1 \leq \alpha_2$
    for each map triangle T_j^t **do**
        Extract the two smallest interior angles $\alpha_{map} \leftarrow \{\beta_1, \beta_2\}$ where $\beta_1 \leq \beta_2$
        Compute angular difference $\delta_\alpha \leftarrow |\alpha_1 - \beta_1| + |\alpha_2 - \beta_2|$
        if $\delta_\alpha < DIFF$ **then**
            Proceed to geometric feature extraction
        end if
    end for
end for

B. Geometric Feature Extraction

For each potential match, extract and analyze geometric features to further verify similarity.

1. Target Triangle Analysis

Extract vertices A, B, C and compute edge vectors $\boldsymbol{AB}, \boldsymbol{BC}, \boldsymbol{CA}$
Identify longest side $\boldsymbol{d}_1$ and calculate centroid $c_t \leftarrow \frac{A+B+C}{3}$
Locate nearest non-vertex landmark $p_{nearest}$ to centroid c_t
Compute reference vector $\boldsymbol{d}_{center} \leftarrow p_{nearest} - c_t$

2. Camera Triangle Analysis

Extract vertices and compute edge vectors
Identify longest side $\boldsymbol{d}'_1$ and calculate centroid c_s
Locate nearest non-vertex landmark to centroid
Compute camera vector $\boldsymbol{d}'_{center}$ from centroid to the nearest landmark

3. Similarity Verification

$\mathcal{I} \leftarrow |\boldsymbol{d}_1 \cdot \boldsymbol{d}_{center} - (\boldsymbol{d}'_1 \cdot \boldsymbol{d}'_{center})/\gamma^2|$ ▷ Inner product discrepancy
$\mathcal{C} \leftarrow |(\boldsymbol{d}_1 \times \boldsymbol{d}_{center}) - ((\boldsymbol{d}'_1 \times \boldsymbol{d}'_{center})/\gamma^2)|$ ▷ Cross product discrepancy
if $\mathcal{I}^2 + \mathcal{C}^2 < DIFF \cdot \|\boldsymbol{d}_{center}\|$ **then**
    Record match in result structure $\mathcal{M}$
end if

III. Transformation Parameter Estimation

This phase calculates the rotation and translation parameters between matched triangles.

A. Rotation Calculation

if $|\mathcal{M}| < 1$ **then**
    return ▷ No matches found, exit algorithm
end if
for each matched triangle pair $(T_i^s, T_j^t) \in \mathcal{M}$ **do**
    Compute rotation angle between corresponding vectors θ_i
end for
Apply statistical filtering: $\theta_{i_{filtered}} \leftarrow \{\theta_i : |\theta_i - \mu_\theta| \leq 3\sigma_\theta\}$
Calculate mean rotation angle ϕ
Construct rotation matrix $R \leftarrow \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{bmatrix}$

B. Initial Translation Estimation

Apply rotation to camera triangle centroids: $c'_S \leftarrow R \cdot c_S$
Compute mean centroid displacement of matched pairs
Calculate spacecraft position $P_{origin} \leftarrow R \cdot [0, 0] + \boldsymbol{T}$

IV. Statistical Outlier Rejection

This phase applies statistical filtering to remove outliers and refine position estimates.

A. Centroid Displacement Analysis

Compute displacement vectors $\Delta c \leftarrow \{c_{t_i} - c'_{s_i} : i \in \mathcal{M}\}$
Calculate mean $\mu_{\Delta C}$ and standard deviation $\sigma_{\Delta C}$ of displacement vectors
Apply 3σ filter: $\mathcal{M}_{3\sigma} \leftarrow \{i \in \mathcal{M} : |\Delta C_i - \mu_{\Delta C}| \leq 3\sigma_{\Delta C}\}$
Apply 2σ filter: $\mathcal{M}_{2\sigma} \leftarrow \{i \in \mathcal{M} : |\Delta C_i - \mu_{\Delta C}| \leq 2\sigma_{\Delta C}\}$

B. Refined Parameter Calculation

Compute mean displacement vectors $\mu_{\Delta C_{3\sigma}}$ and $\mu_{\Delta C_{2\sigma}}$
Calculate 3σ-filtered position estimate: $P_{3\sigma} \leftarrow R \cdot [0, 0] + \mu_{\Delta C_{3\sigma}}$
Calculate 2σ-filtered position estimate: $P_{2\sigma} \leftarrow R \cdot [0, 0] + \mu_{\Delta C_{2\sigma}}$

V. Reliability Assessment

This phase evaluates the reliability of the position estimate and computes final metrics.

Calculate inlier ratio: $\rho_{inliers} \leftarrow \frac{|\mathcal{M}_{3\sigma}|}{|\mathcal{M}|}$

Compute position difference: $\delta_{origin} \leftarrow \|P_{3\sigma} - P_{2\sigma}\|$

Evaluate spread of centroid displacements: $\sigma_{spread} \leftarrow |\mu_{\Delta C_{3\sigma}} - \mu_{\Delta C_{2\sigma}}|$

if $(0.5 \cdot \rho_{inliers} + 0.25 \cdot \delta_{origin} + 0.25 \cdot \sigma_{spread}) \leq T_{rel}$ **then**

    $reliability \leftarrow 1$ ▷ Position estimate is consistent

else

    $reliability \leftarrow 0$ ▷ Position estimate is inconsistent

end if

Provide final estimated position $P_{final} \leftarrow P_{3\sigma}$

Appendix B

Appendix - RANSAC Algorithm Pseudo-code

Algorithm 2 Neighborhood Matching Algorithm (RANSAC) Pseudo-code

I. Data Acquisition and Preprocessing
This phase retrieves image and reference data. Structures and thresholds for the matching are initialized.

A. Data Retrieval
1: Retrieve histogram set H_t from reference catalog
2: Obtain landmark map from source image
B. Parameter Initialization
3: Configure similarity thresholds: $k \leftarrow 7$, $m \leftarrow 5$, T $\leftarrow 0.7$, $n_b \leftarrow 4$, $T_c \leftarrow 5 \cdot 10^4$, $T_r \leftarrow 1 \cdot 10^3$, $n_s \leftarrow 4$
4: Initialize parameters and data structures

II. Neighborhood and Histogram Construction
for each landmark i in source image landmarks **do**
    A. Detect neighbors:
    Find all neighbors within radius $k \cdot r_c$ and sort by size
    if insufficient neighbors ($< m$) **then**
        continue ▷ Skip landmarks with too few neighbors
    end if
    Compute reference vector $\boldsymbol{v}_1$ to largest neighbor

    B. Extract features for each neighbor:
    for each neighbor j (except the largest) **do**
        Compute vector $\boldsymbol{v}_j$ from center to neighbor
        $f_1 \leftarrow$ normalized angle between $\boldsymbol{v}_j$ and $\boldsymbol{v}_1$ (range [0,1])
        $f_2 \leftarrow$ radius ratio r_c/r_j, $f_3 \leftarrow$ distance ratio $\|\boldsymbol{v}_j\|/\|\boldsymbol{v}_1\|$
        Store feature vector $[f_1, f_2, f_3]$ for this neighbor
    end for
    Normalize f_2 and f_3 to range [0,1] using min-max normalization

    C. Create histogram
    Create bin edges and initialize histogram H
    for each feature vector **do**
        Find appropriate bins for f_1, f_2, f_3 and increment histogram count
    end for
    Store histogram: Hs[i] $\leftarrow H$
end for

III. Histogram Matching
for each camera landmark histogram Hs[i] **do**
    for each map landmark histogram Ht[j] **do**
        Compute histogram Euclidean distance between Hs[i] and Ht[j]
        if histograms match (distance $<$ T) **then**
            Store camera and map landmark data with reference vectors
            break ▷ Move to next camera landmark
        end if
    end for
end for

IV: Coarse to fine Transformation Estimation
if sufficient histograms correspondences (≥ 1) **then**
    for each correspondence i **do**
        A. Compute coarse transformation
        Extract points $\boldsymbol{c}_t$, $\boldsymbol{c}_s$ and compute scale factor s
        Compute rotation angle ϕ between source and target landmark pair
        Create rotation matrix $\boldsymbol{R}$ and obtain translation vector $\boldsymbol{t}$
        Transform all source points using s, $\boldsymbol{R}$, $\boldsymbol{t}$ to get $\boldsymbol{c}'_{s_i}$

        B. Compute fine transformation
        for RANSAC iterations 1 to n_s **do**
            Randomly select 4 correspondences of $\boldsymbol{c}_{t_i}$ and $\boldsymbol{c}_{s_i}$
            Set up linear system $\boldsymbol{Bh} = \boldsymbol{Y}$ for homography parameters
            Solve for homography parameters $\boldsymbol{h}$
            Apply homography to all points and compute cost function
            Identify inliers (singular cost $< T_c$)
            if current transformation has lower cost than best so far **then**
                Update best transformation parameters and inlier set
                Update best cost
            end if
        end for
    end for

    V. Position Estimation and Reliability Assessment
    if inliers exist **then**
        Get target and source data for inliers
        Compute average displacement between matched landmarks
        if inliers cost $< T_r$ **then**
            Reliability $\leftarrow 1$ ▷ Position estimate is consistent
        else
            Reliability $\leftarrow 0$ ▷ Position estimate is inconsistent
        end if
        Compute spacecraft position
        Transform origin using selected similarity and homography
    end if
else
    Report insufficient correspondences for matching
end if

Bibliography

[1] Clara O'Farrell et al. «ASPIRE2: The Mars Sample Retrieval Lander's Supersonic Parachute Test Program». In: *2023 IEEE Aerospace Conference*. 2023, pp. 1–9. DOI: 10.1109/AERO55745.2023.10115818.

[2] Chinmayee Chaini and Vijay Kumar Jha. «A review on deep learning-based automated lunar crater detection». In: *Earth Science Informatics* 17.5 (Oct. 2024), pp. 3863–3898. ISSN: 1865-0481. DOI: 10.1007/s12145-024-01396-2. URL: https://doi.org/10.1007/s12145-024-01396-2.

[3] Bertrand Leroy et al. «Crater detection for autonomous landing on asteroids». In: *Image and Vision Computing* 19.11 (2001), pp. 787–792.

[4] Yang Cheng and A. Ansar. «Landmark Based Position Estimation for Pinpoint Landing on Mars». In: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. 2005, pp. 4470–4475. DOI: 10.1109/ROBOT.2005.1570808.

[5] Leena Singh and Sungyung Lim. «On Lunar On-Orbit Vision-Based Navigation: Terrain Mapping, Feature Tracking Driven EKF». In: Aug. 2008. ISBN: 978-1-60086-999-0. DOI: 10.2514/6.2008-6834.

[6] Chad Hanak, Tim Crain, and Robert Bishop. «Crater Identification Algorithm for the Lost in Low Lunar Orbit Scenario». In: Feb. 2010.

[7] Vincent Simard Bilodeau. «Autonomous vision-based terrain-relative navigation for planetary exploration». PhD thesis. Ph. D. Thesis, Sherbrooke Univ., Quebec, Canada, 2015.

[8] Bolko Maass et al. «Crater Navigation System for Autonomous Precision Landing on the Moon». In: *Journal of Guidance, Control, and Dynamics* 43 (Apr. 2020), pp. 1–18. DOI: 10.2514/1.G004850.

[9] Mingda Jin and Wei Shao. «Crater Triangle Matching Algorithm Based on Fused Geometric and Regional Features». In: *Aerospace* 11.6 (2024), p. 417.

[10] Fumito Uwano et al. «Theoretical Analysis of Triangle Matching Method Based on Craters for Spacecraft Localization». In: ().

[11] Yuan Li and Yuyang Zeng. «Lunar Image Matching Based on Impact Crater Morphology and Distribution for Landing Localization». In: *IGARSS 2024-2024 IEEE International Geoscience and Remote Sensing Symposium*. IEEE. 2024, pp. 6146–6149.

[12] Iain Martin, Martin Dunstan, and Manuel Sanchez Gestido. «Planetary surface image generation for testing future space missions with pangu». In: *2nd RPI Space Imaging Workshop*. Sensing, Estimation, and Automation Laboratory. 2019.

[13] Zhengshi Yu, Pingyuan Cui, and John L. Crassidis. «Design and optimization of navigation and guidance techniques for Mars pinpoint landing: Review and prospect». In: *Progress in Aerospace Sciences* 94 (2017), pp. 82–94. ISSN: 0376-0421. DOI: https://doi.org/10.1016/j.paerosci.2017.08.002. URL: https://www.sciencedirect.com/science/article/pii/S0376042117301082.