SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

A local exploration cost function for removing alignment constraints in perception-aware path following

Hannah Makarenko

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

A local exploration cost function for removing alignment constraints in perception-aware path following

Ein lokale Erkundungskostenfunktion zur Aufhebung von Ausrichtungsbeschränkungen für wahrnehmungsbewusstes Pfadverfolgen

Author:	Hannah Makarenko
Supervisor:	Moritz Kuhne, Florian Steidle
Advisor:	Alin Olimpiu Albu-Schäffer
Submission Date:	22.06.2024

I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Munich, 22.06.2024 Hannah Makarenko

Abstract

The research field of space exploration is growing and plays a crucial role [9] in replacing human presence, as manned missions to Mars are dangerous and time-consuming. The Deutsche Luft- und Raumfahrt RMC research group is tasked with building a prototype of a mobile Lightweight Rover Unit (LRU) for autonomous navigation, as communication with Earth would result in an unbearable response delay during remotely controlled exploration. At the current state, the rover is able to find a path to a predefined target and reach it safely using visual perception from a pan-tilt camera to detect obstacles along the path, and recalculating the path if necessary. To do so, the camera always follows the path ahead, thereby ignoring to collect important information of the environment alongside the path.

The goal of this study is to extend visual exploration of the LRU's environment to collect more data about the Martian surface. Our approach incorporates additional degree of freedom of the camera in controlling the pan angle while following a predefined path. The challenge is to maintain sufficient focus on the path to avoid collisions or traveling through unknown terrain in autonomous navigation, while allowing enough freedom for visual exploration. Recent work in autonomous exploration has focused on finding the most efficient path to cover all the unknown surroundings, given the current knowledge of the map, to explore the entire environment. However, our work differs from conventional approaches, since the target position in our scenarios is predefined. Therefore, our objective is to extend the visual exploration along the path towards the specified goal.
To find the best pan orientation, we introduce a simplified simulation of the visual perception and traversal through the world map. In order to evaluate and compare different approaches we examine them for the following metrics: (1) whether a collision occurred or the rover passed through unseen terrain, (2) the length of the path traveled, (3) the total exploration coverage along the path, (4) and the time required to determine the pan orientation. We sample over candidate pan angles and compare the potential seen costs, defined here as inverse reward function from information gain that the camera would capture in the travel environment. We aim to identify the best cost function for selecting best-suited pan orientation from the following: (1) static pan orientation aligned to the rover's body, which does not utilize the full range of pan

motion of the camera and therefore not necessitate in a cost function; (2) a cost function resulting in camera following the path ahead; (3) a cost function, inspired by Boyu Zho et al. [1], incorporating the lateral distance to the path in combination with the longitudinal distance relative to the rover's current position; (4) and a cost function prioritizing the closest distance to the boundary of unknown areas. We show that the proposed cost function based on previous work of Boyu Zho et al. [1] is the best-suited strategy to allow visual exploration of the environment while ensuring a safe travel.

Contents

1 Introduction

The research field of space exploration is growing and aims to expand our knowledge of the universe. The use of mobile robots that can move autonomously on extraterrestrial surfaces plays a crucial role [9] for replacing human presence, as manned missions to Mars are dangerous and time-consuming. The DLR-RMC research group is tasked with building the mobile Lightweight Rover Unit (LRU) to explore Mars in future expeditions. This robot is a prototype for research applications in autonomous navigation, as communication with Earth would result in an unbearable response delay during remotely controlled explorations. The rover is a ground based robot equipped with a pan-tilt camera unit for visual perception [9].

Figure 1.1: LRU2 during the ROBEX demo mission space campaign that took place during June-July 2017 on Mt. Etna, Italy [9]

In its current development stage, the LRU autonomously navigates to a given target by sensing its environment with a pan-tilt camera. Using computer vision, it detects obstacles, allowing the path planning unit to adjust the path and plan an obstacle-free route from the current location to the specified target. For the robot to reach its target safely, the camera movement must meet specific requirements, like aligning with the planned path's direction to enable early obstacle detection to avoid collisions and potential damage or safety hazards. Additionally, the system must avoid passing unknown

areas that have not been previously captured by the camera. Currently, this is achieved with a camera following the path ahead. However, by not utilizing the camera's full range of motion the system misses out on versatile exploration of the environment surrounding the original path.

Since autonomous driving is currently a highly researched field and easily understood by a large part of the population, we would like to briefly compare it to autonomous systems and highlight the differences. Both fields involve sensing the environment through visual perception, localization and mapping, and planning. However, autonomous extraterrestrial exploration differs from autonomous driving due to the lack of knowledge about the environment. In autonomous driving, the challenge is to detect and react to objects such as traffic lights, other moving cars, or pedestrians, while following a pre-calculated path, based on Global Navigation Satellite Systems (GNSS), in a previously partially known environment. On the other hand, autonomous systems do not operate in a known and structured environment such as roads, and therefore must explore their environment to reach their target and continuously adapt the path toward it, like when obstacles are detected.

1.1 Goals and objectives

Our goal is to develop a strategy for the camera's panning motion that combines the goal reaching task with the exploration of the environment along the planned path. To achieve this, we create a simplified two-dimensional simulation of the discretized world in a grid map, implying free cells for traversal or blocked cells with obstacles. We introduce cost functions that allow us to assign scalar values based on different criteria, i.e., distance to the path, current position of the LRU, or whether this area has already been captured by the camera. This allows us to determine the next best pan orientation for the camera at each path point. In addition to the previously mentioned hard requirements, avoiding collisions and passing through unknown areas, the cost functions are examined for the following soft requirements: total exploration of the environment along the planned path, length of the traversed path, and average time for selecting a pan angle at each path point. Therefore, we will compare the currently implemented static forward pan alignment with different cost functions: first, one that follows the path ahead; second, one inspired by B. Zho. [1], which includes both the lateral distance to the path and the longitudinal distance relative to the rover's current position. The third cost function, where the camera faces the nearest boundary of unknown terrain and prioritizes the shortest distance. This results in a behavior where

the camera is always pointed at the nearest boundary of the unknown area.

The goal of this thesis is an algorithm for controlling the camera's pan angle that expands the area the rover observes as it follows a path to a predefined destination. Thereby the camera must focus on the path to avoid collisions or traversing unknown areas. Currently, the mounted pan-tilt camera is constrained to point in the direction of the rover's path. Our main objective is to develop an approach that allows the LRU to perceive its local environment more comprehensively along its planned path by exploiting the degrees of freedom of the pan-tilt camera, and thus the ability to move within the given direction. To achieve this behavior, we use a cost function to determine the best orientation angle for the camera that allows it to deviate from the direction of the planned path. To find the most appropriate cost function for this task, we want to compare several cost functions.

1.2 Outline

The remainder of this thesis is structured as follows:

In Chapter 2, we introduce the terminology and background knowledge needed to understand the concepts of visual perception in computer vision. This includes the pinhole model and the transformations from world coordinates to the position and alignment of the camera mounted on the LRU.

In Chapter 3, we survey the related work done in this area and provide a description of the approach relevant to this thesis.

Next, in Chapter 4, we present our proposed methods, which includes the explanation of the sampling process, a detailed exploration of the simulation we are using, a mathematical description of the analyzed cost functions, and the metrics used for comparison.

Chapter 5 examines the implementation of our cost functions, followed by an analysis and discussion of the results in Chapter 6.

Finally, in Chapter 7 we provide a conclusion for the thesis and an outlook on further work that can be built on the presented results.

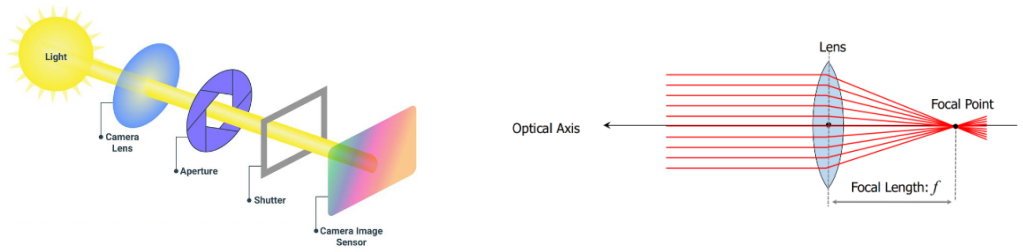
2 State of the Art

How do humans perceive their environment through their eyes? Reflected light from objects passes through the lens of the human eye, converging and projecting onto the retina at the back of the eye. This creates a two-dimensional projection of the three-dimensional reality, resulting in some information loss. However, the human brain reconstructs this reality using contextual clues. For example, the size and prior knowledge of an object can provide information about its distance and position in space. Shadows and colors can also offer insights into the surface and three-dimensional structure of objects. Although humans can only perceive a two-dimensional projection of their three-dimensional environment, we are still able to interpret our surroundings through additional cognitive processing.
The process of perception can be easily imitated with a pinhole camera, but interpreting the perceived information is much more complex and requires significant computational effort to reconstruct the original three-dimensional form and spatial position. This is the essence of what Computer Vision aims to achieve. The following sections explain the pinhole camera model and the transformations that project 3D world points onto the image plane.

2.1 Pinhole Camera Model

To understand the fundamentals of Computer Vision it is crucial to understand the basics of a pinhole camera. As illustrated in Figure 2.1a, light enters the camera through the lens, which converges the incoming light. It then passes through the aperture, which regulates the amount of light entering the camera. The shutter speed determines how long the film is exposed to this light.

The converged light intersects at the focal point, projecting the captured object inverted onto the camera's image plane. The distance between the lens and the focal point, known as the focal length, is crucial for further explanations. 2.1b



(a) From Light Signal to Electrical Signal

(b) Converging Lens

Figure 2.1: The basic principles of a pinhole camera illustrate the different layers that influence the light before it is inverted and projected onto the image plane. [12]

2.2 Transformations - from 3D world points to 2D image

The transformation of $3D$ world coordinates into $2D$ image pixels is given by the equation:

$$\lambda p = K \cdot [R|t] \cdot P^W \tag{2.1}$$

where P^W is a $3D$ point in the world coordinate system and p is the corresponding $2D$ pixel in an imaginary image plane [17]. The following sections will describe the transformation equation from right to left for better understanding.

2.2.1 Extrinsic Matrix

The extrinsic matrix describes the transformation from the origin of the world coordinate system to the position of the camera coordinate system. In other words, it describes the position and alignment of the camera in space, as shown in 2.2. This includes the rotation R and the translation t and can be expressed as $g(x) = Rx + t$, where $R \in \mathbb{R}^{3\times3}$ and $t \in \mathbb{R}^3$.

Translation violates the property of linear transformation because it shifts the origin. To address this, we use homogeneous coordinates [10], where $x \in \mathbb{R}^3$ is represented as:

$$\begin{pmatrix} x \\ 1 \end{pmatrix} \in \mathbb{R}^{n+1} \tag{2.2}$$

This allows $g(x)$ to become a linear mapping, defined as $g : \mathbb{R}^{n+1} \mapsto \mathbb{R}^{n+1}$, with the transformation:

$$\begin{pmatrix} x \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix} \tag{2.3}$$

Thus the transformation from world coordinates to camera coordinates can be expressed as [7]:

$$P^C = \begin{bmatrix} X^C \\ Y^C \\ Z^C \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X^W \\ Y^W \\ Z^W \\ 1 \end{bmatrix} \tag{2.4}$$

In the specific case of the LRU, we need a transformation chain as shown below:

$${}_{\mathrm{C}}T^{\mathrm{W}} = {}_{\mathrm{C}}T^{\mathrm{base}} \cdot {}_{\mathrm{base}}T^{\mathrm{tcp}} \cdot {}_{\mathrm{tcp}}T^{\mathrm{r}} \cdot {}_{\mathrm{r}}T^{\mathrm{W}} \tag{2.5}$$

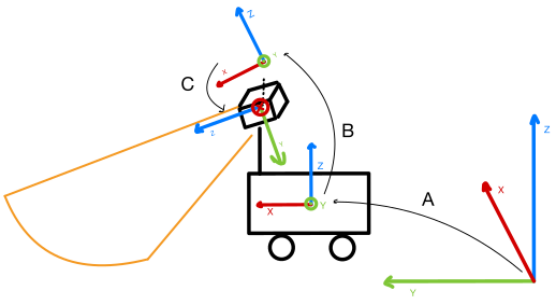


Figure 2.2: The sequence of transformations from the world origin coordinate system to the camera coordinate system of the LRU, with the camera frustum depicted in orange. The coordinate system axes are colored as follows: x-axis in red, y-axis in green, and z-axis in blue. The transformation steps, labeled A, B, and C, are detailed further in 2.2.1

where ${}_{\mathrm{r}}T^{\mathrm{W}}$ performs the translation from the world coordinate origin to the rover's position and its body yaw alignment (A in Figure 2.2) given by x, y and θ. ${}_{\mathrm{tcp}}T^{\mathrm{r}}$ is a static transformation from the LRU body to tool center point of the mounted camera, followed by the pan-tilt rotation with ${}_{\mathrm{base}}T^{\mathrm{tcp}}$ (B in Figure 2.2). The last static transformation ${}_{\mathrm{C}}T^{\mathrm{base}}$ aligns the camera to the desired direction, where the Z-axis points out of the picture (C in Figure 2.2).

2.2.2 Intrinsic Matrix

After the position and alignment in space of the camera is set by the extrinsic matrix we need to map the world points onto the camera's image plane. Therefore, we introduce the intrinsic matrix which maps the $3D$ points from the camera coordinate system to $2D$ pixels of an imaginary image plane and can be expressed as [17]:

$$\lambda p = K \cdot P^{\mathrm{C}} \tag{2.6}$$

$$\text{with } K = K_s \cdot K_f \cdot \Pi_0 \tag{2.7}$$

where we have introduced three new matrices:

$$K_s = \begin{bmatrix} 0 & 0 & o_x \\ 0 & 0 & o_y \\ 0 & 0 & 1 \end{bmatrix}, K_f = \begin{bmatrix} f_x & 1 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{bmatrix}, \Pi_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.8}$$

The 3×4 matrix Π_0, known as standard projection matrix, is essential for transforming a 4 dimensional point in camera coordinates to 3 dimensions. The focal length between the focal point and the imagenary image plane is defined by K_f, followed by K_s which determines pixel coordinates of the image. If the camera is not centered at the optical center we can compensate this with o_x and o_y. [17].

Putting the extrinsic and intrinsic matrices together we obtain the general projection matrix Π [17]:

$$\lambda p = \begin{bmatrix} 0 & 0 & o_x \\ 0 & 0 & o_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \cdot P^W = K \cdot \Pi_0 \cdot gP^W = \Pi \cdot P^W \tag{2.9}$$

2.2.3 2D Mapping

In summary, we have introduced a series of matrix multiplications that result in a $3D$ pixel point (2.10). The final step is to convert this into a $2D$ pixel plane. [8]

$$\begin{bmatrix} X^W \\ Y^W \\ Z^W \\ 1 \end{bmatrix} \xrightarrow{\text{extrinsic Matrix}} \begin{bmatrix} X^C \\ Y^C \\ Z^C \\ 1 \end{bmatrix} \xrightarrow{K_f, \Pi_0} \text{image coordinates} \xrightarrow{K_s} \text{pixel coordinates} \tag{2.10}$$

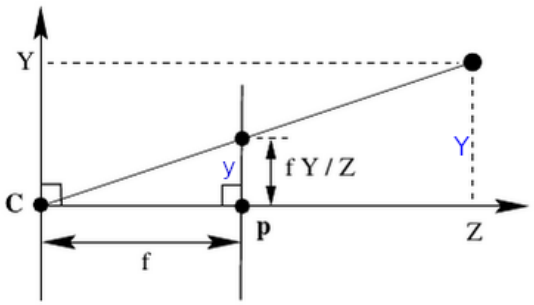


Figure 2.3: Illustration of triangulation [8]

For simplicity reasons we can consider the image plane to be in front of the center of projection, rather than behind it. As shown in 2.3, we obtain a triangulation of

$$\tan(\alpha) = \frac{y}{f} = \frac{Y}{Z} \Longrightarrow y = \frac{fY}{Z} \tag{2.11}$$

To obtain the $2D$ pixel coordinate $(u, v)^{\mathrm{T}}$ we can divide the equation by the scaling factor λ,

$$u = \frac{{\pi_1}^{\mathrm{T}} P^W}{{\pi_3}^{\mathrm{T}} P^W}, \; v = \frac{{\pi_2}^{\mathrm{T}} P^W}{{\pi_3}^{\mathrm{T}} P^W} \tag{2.12}$$

where ${\pi_1}^{\mathrm{T}}$, ${\pi_2}^{\mathrm{T}}$, ${\pi_3}^{\mathrm{T}} \in \mathbb{R}^4$ are the three rows of the general projection matrix Π [17].

3 Related Work

Autonomous exploration seeks to expand the understanding of the surrounding environment. The general goal of exploration is to perceive an a priori unknown environment and map the surroundings. There, the fundamental challenge in exploration is to determine the best subsequent movement of the robot, given the current knowledge of the environment, in order to cover the observation most efficiently, that means as comprehensive as possible and as quickly as possible. Classical autonomous exploration methods mainly include frontier-based [16] and information-theoretical [5] strategies. The former approach describes a semantic strategy where frontiers are regions on the boundary between open space and unexplored space. The goal is to move these boundaries in order to increase the knowledge about the unseen territory. The frontier-based strategy will eventually stop, when all accessible areas are explored [16]. Another approach is the information-theoretic approaches [2] as Next Best View (NBV) [3]. Here, the world is rasterized in a $3D$ voxel map, such that a robot can progressively discover its surroundings during its mission by focusing on expanding the known volume [4].

While our primary goal is autonomous navigation to a specific target, we also aim to maximize exploration of the environment along the route. This differentiates our study from traditional exploration tasks. In classic exploration, robots strive to map unknown areas using the most efficient path. However, we have a predefined path to reach the target. The challenge lies in extending visual exploration capabilities along this predetermined path, enabling us to collect additional environmental information while maintaining efficient navigation toward the goal.

The authors of **RAPTOR: Robust and Perception-Aware Trajectory Replanning for Quadrotor Fast Flight** [1] address the issue of maximizing environmental perception along a given path towards a specified goal position. For a quadcopter with a camera fixed to its body orientation, they plan yaw angles along the given path. We adapt their methodology by incorporating information gain based on the lateral distance to the path and the longitudinal distance to the rover into a cost function. A more detailed explanation of this approach is presented in section 4.3.
Although the objectives of exploration intersect in both projects of RAPTOR [1] and

the LRU, there are some differences. In the context of high-speed quadrotors, fast adaptability is required to ensure safety, which necessitates a greater focus on the path. The high speed requires more frequent adjustments to the planned path and increased caution when encountering unexpected obstacles. In our case, the rover is comparatively slow, with a speed of 4 km/h [9], resulting in a more comprehensive exploration of the environment.

In our experiment, we introduce a cost function method similar to the classical frontier-based approach [16]. We aim to formulate a semantic concept of a camera moving to the closest boundary of unknown terrain into a cost function, ensuring it captures unseen areas ahead before traversing them. Although this strategy resembles the frontier-based approach, our objective differs and cannot be directly adapted. Previous research aimed to explore the entire environment, while our goal is to follow a predefined path and expand visual exploration along it.

4 Methodology

After establishing the basic terminology and knowledge required, we proceed to describe our proposed methods. In the following section, we introduce the simulation environment and describe the sampling method used in our experiment. We then explain the mathematical formulations of the cost functions and the metrics used for comparison.

4.1 Simulation Environments

In robotics research, simulation is an important component because it allows us to develop in a lightweight and safe test environment. To test the proposed methods, we simulate the path following and exploration task. To do this, we use `numpy.arrays` that represent the world in which the rover operates, marking areas it has already observed with 1 and unknown areas with 0. Using this simplified simulation, we can recreate the camera view and the projection of the environment. In addition, we can simulate the movement of the LRU through the simulated map. However, we do not consider the dynamics of the system in this simulation, since we are only interested in the projected view, and we assume that the rover is moving slower, comparable to the angular velocity of the camera.

4.2 Sampling Approach

On the LRU, sampling is used for planning over motions in the combined states of body and pan-tilt camera, and is therefore predefined by the system. The use of sampling allows the formulation of different cost functions by adding the resulting cost maps into one, so the sampling approach is extensible without limitations. Other reasons why sampling is beneficial: it allows formulating cost functions without restrictions on differentiability.
Therefore, in this thesis, we evaluate candidate cost functions over sampled pan angles as this approach integrates well into the LRU planning solution. In this thesis, we consider only the panning motion of the camera because including the tilt angle would lead to an exponential increase in the number of test cases. Therefore, we limit our

objective to the pan angle, but the experiment can be adapted analogously for the tilt angle analysis.

For the sampling method, we need to rasterize the world into a grid, which we convert into a cost map. To construct a cost map, we use the cost function to calculate the information gain that the rover would gain if it captured that position for each map cell based on the rover's current position or path, and assign that value to the cost map. The samples represent potential discretized pan angles for each point along the path. For each view orientation, we accumulate the values of all visible points from the cost map into a single scalar, allowing us to compare all samples at each path point. This process allows us to prioritize and select the pan orientation with the lowest observed cost.

4.3 Cost Functions

Each pan sample is evaluated using a cost function. These cost functions act as inverse reward functions, where a lower cost indicates a higher potential information gain. In other words, the cost function estimates how much new information the camera would gather if it were pointed towards a specific region based on the pan angle. The following subsections explain the mathematical formulations of the implemented cost function methods.

4.3.1 Static Pan Alignment

This experiment demonstrates a scenario in which the degrees of freedom of the attached pan-tilt camera are not utilized, emphasizing the necessity of camera movement during the path pursuit. In this case, we do not use a cost function nor samples over pan alignments; instead, the pan angle is aligned with the yaw orientation of the LRU's body and a tilt angle of 0°, ensuring the camera faces the direction of the rover.

4.3.2 Follow Path Pan Alignments

In this method, we calculate the deviation of the pan samples from an angle α, which is defined as the angle from the rover's current position to a point on the path, and prioritize the minimum deviation. In other words, this cost function results in the pan-tilt camera movement tracking the path ahead without deviating to the surrounding areas along the path. Only the knowledge of the path is required for its calculation. Thus, the relation can be expressed as:

$$\min_{\beta \in pan\_samples} = |\alpha - \beta| \tag{4.1}$$

4.3.3 Cost Function with Lateral and Longitudinal Distance to Path

This cost function is inspired by the previously mentioned paper [1]. This method considers both the lateral distance to the path and the longitudinal distance to the current position of the LRU to determine the information gain (Ig) for each map cell. The cost function for a map point therefore requires both information about the path and information about known/unknown areas, so the total cost function for a pan angle can be represented as:

$$\min_{\beta \in pan\_samples} = \sum_{p \in seen\_points(\beta)} cost(p) \tag{4.2}$$

with $seen\_points(\beta)$ as the set of all projected image points for the pan angle β and the cost for each position p on the map as:

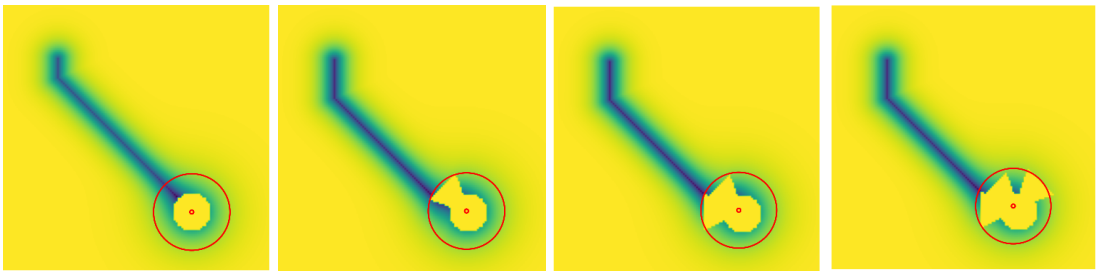
$$cost(p) = 2 - Ig(p) = 2 - [\exp(-\lambda_{lat} \cdot d(p, path)) + \exp(-\lambda_{long} \cdot d(p, rov\_pos))] \tag{4.3}$$

where $d(p, path)$ is the distance from p to the closest point on the path, $d(p, rov\_pos)$ is the distance from p to the rover's current position and λ_{lat} and λ_{long} are coefficients for the lateral and longitudinal distance and $seen\_points$ is a set of all observed world map points captured within each pan sample.

We aim to minimize costs for more interesting areas while maximizing costs for less interesting areas. This ensures that the rover explores its environment more comprehensively when it is in already known areas and is forced to follow its path upon reaching the boundary of unknown terrain. As $\lim_{x \to 0} \exp(-x)$ converges to 1 for closer regions, we obtain maximal information gain and, consequently, minimal cost. Conversely, as $\lim_{x \to \inf} \exp(-x)$ converges to 0 for regions further away from the LRU, we achieve minimal information gain and maximal cost. Using two parameters for information gain calculation, we can conclude that $Ig \in [0, 2]$. By subtracting Ig from a value of 2, we transform the information gain into a cost function.

After Ig and $cost(p)$ are calculated for each point on the map, the resulting cost map is masked with known areas, assigning these areas a value of 2. This assignment

represents the maximum cost for already observed areas, rendering them uninteresting for the next pan alignment and prioritizing unseen areas. For better understanding a cost map is visualized in 4.1



(30,40) (31,41) (32,42) (33,43)

Figure 4.1: Segments form exploration for Scenario 2 at positions (30,40) to (33,43): cost maps with color range from blue as the lowest cost (maximal information gain) to yellow as the highest cost (minimal information gain) for current position (denoted red point and red circle to visualize the LRU's range of view)

The coefficients λ_{lat} and λ_{long} allow us to weight the information gain for lateral and longitudinal distances. A coefficient closer to 0 maximizes the Ig and simultaneously minimizes the $cost$, while a coefficient $\lambda > 1$ minimizes the Ig and accordingly maximizes the $cost$. In our case we choose $\lambda_{lat} = 0.2$ and $\lambda_{long} = 0.1$ to prioritize the longitudinal distance to the rover's current position, thereby achieving more comprehensive exploration.

4.3.4 Cost Function with Distance to Border

The last cost function calculates for each map cell p the distance to the closest border of unknown terrain and can be expressed with:

$$\min_{\beta \in pan\_samples} = \sum_{p \in seen\_points(\beta)} d(p, b) \tag{4.4}$$

where $seen\_points(\beta)$ is the set of all projected image points for the pan angle β and $d(p, b)$ is the distance from the observed world point to the closest border point b of the

unknown area. This results in a behavior where the LRU prioritizes the pan sample that points to the nearest boundary of the unknown to ensure that the LRU never traverses unseen terrain. Therefore, this function only requires information about the known and unknown map.

4.4 Metrics

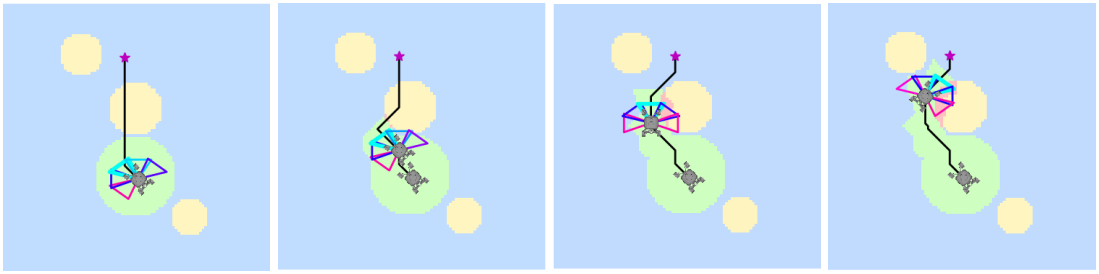
For the analysis and comparison of the proposed cost functions, we introduce the following metrics:

- occurrence of collision
- traversed path length (in path points)
- total map exploration (in map grid points)
- average time for cost function calculation

Avoiding collisions and traversing unknown areas are essential requirements for the autonomous navigation task, so it is important to evaluate this in our simulation. For example, if the camera does not examine the area in front of the rover thoroughly enough, a collision could occur. In our experiment we treat passing through unknown terrain as collisions to ensure that potential crashes are avoided.
It would be interesting to see if the path length changes with different cost function strategies, as the cost function would lead to a different route and thus change the length of the traveled path. Since we are using the A^* algorithm for the path finding task, the order in which the camera would capture the environment could affect the subsequent path recalculation, since known areas are weighted less for the A^* heuristic.
Total map exploration is the essential metric for our experiment, as we aim to expand the observed areas while achieving a given goal.
The last time measurement for the cost map computation shows whether the proposed method is able to run online in the system.

4.5 Testing

For better debugging, we visualize the output of our simulation using plots generated by the `matplotlib.pyplot` interface, inspired by plot-driven development as presented in [11]. These resulting plots can be converted into an animation `.gif` that simulates the path following process, incorporating the cost function to determine the best camera



(a) (b) (c) (d)

Figure 4.2: Simulation fragments for Scenario 0 include 5 pan alignment samples. Further details regarding the color coding can be found in 4.5

pan orientation.

Figure 4.2 presents segments from the simulation animation in which the black line represents the planned path from the LRU's current position to the goal, marked by a star. Blue areas indicate free spaces available for exploration, while obstacles are depicted in yellow. Green fields denote free areas observed by the LRU, and red areas represent observed regions that are blocked by obstacles. As illustrated in figures 4.2b, 4.2c and 4.2d, the path is continuously adjusted as the rover acquires more information about its surrounding environment. The triangles attached to the LRU's body indicate samples of potential pan angles, with corresponding costs for each alignment, color-coded from pink (indicating maximal cost) over dark blue and turquoise (indicating minimal cost). The selected sample with minimal cost is highlighted with a bold triangle. In our static experiment, we do not consider variations in pan angles. Instead, we represent the pan orientation with a simple orange triangle.
The rover's behavior is evaluated across eleven generated scenarios, as shown in 4.3 with varying numbers of pan samples, 5 and 15, to introduce more variability. Scenario 0, Scenario 1 and Scenario 2 are generated maps with randomly placed large obstacles. Scenario 3 to Scenario 7 are maps of existing cities (Boston, Denver, London, New York, Shanghai) [13] with many small obstacles to introduce a variety of test cases. The last three scenarios, Scenario 8, Scenario 9 and Scenario 10 utilize maps derived from real-world data collected during the ROBEX demonstration mission space campaign. This campaign, held in June-July 2017 on Mount Etna, Italy [14], aimed to simulate Martian exploration by using a volcanic landscape that closely resembles the Martian surface. These maps allow us to adapt our simulation to a more realistic Martian environment.

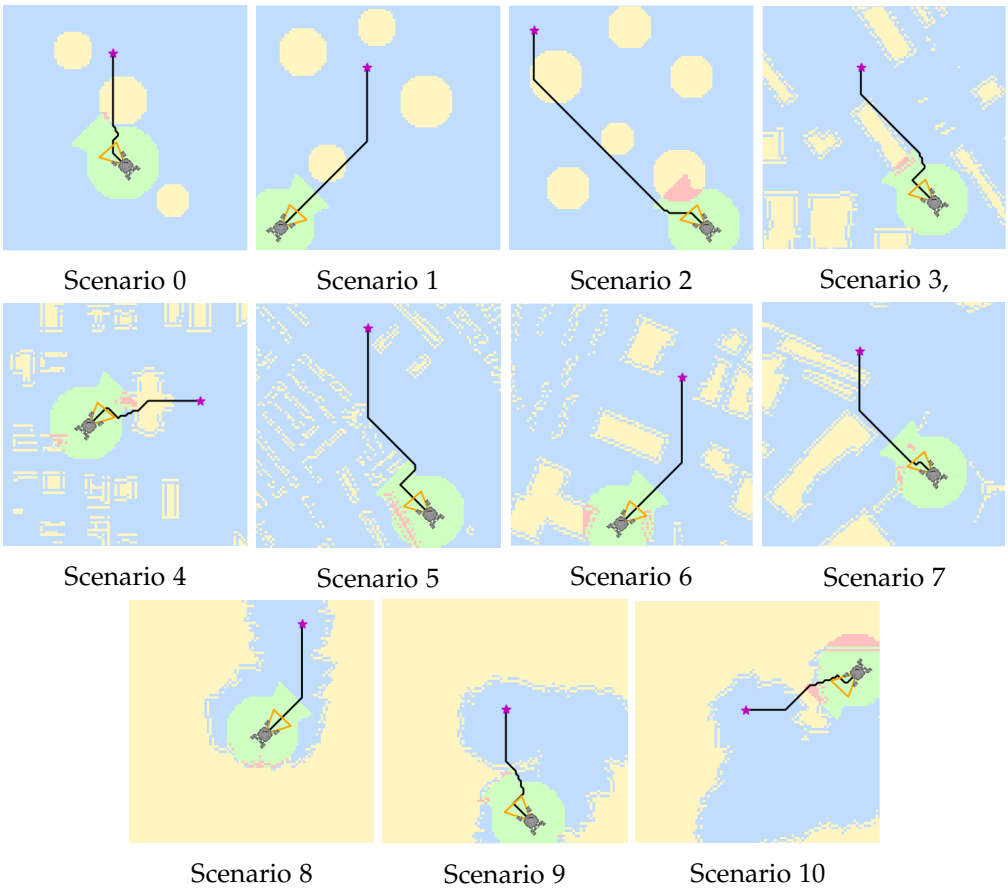


Figure 4.3: Scenarios 0 to 2: randomly generated maps with large obstacles; scenario 3 to 7 are maps of existing cities (Boston, Denver, London, New York, Shanghai) with many small obstacles to introduce a variety of test cases [13]; Scenario 8 to 10 show maps derived from real-world data collected during the ROBEX demo mission space campaign that took place on Mt. Etna, Italy [14]

5 Implementation

This section presents the simulation implementation of the camera's projection of the world and its traversal through the map. It includes the overall code structure and the implementation of the cost functions.

5.1 Camera Simulation

This simplified pseudocode represents the iteration through the path points toward the predefined goal. At each point, candidate pan angle samples are simulated to determine the best pan orientation. The general simulation process can be described as follows:

```
function EXPLORE_PATH(goal, known_map, current_position, pan, tilt, [pan_samples])
    simulated_costs[] = empty_array()
    simulated_points[] = empty_array()
    known_obstacle_map = zeros_like(known_map)
    traversed_path = empty_path()
    path_to_goal = recalculate_path(goal, current_position, pan, tilt, known_obstacle_map)
    while path_to_goal not empty do
        for panin[pan_samples] do
            seen_points, seen_costs = simulate_camera_view(goal, known_obstacle_map, current_position,
            pan, tilt)
            simulated_costs.append(seen_costs)
            simulated_points.append(seen_points)
        angle = determine_min_cost(simulated_costs)
        known_obstacle_map = update_map()
        plot_path_point()
        current_position += 1
        path_to_goal = recalculate_path(goal, current_position, angle, tilt, known_obstacle_map)
    return
```

The function $recalculate\_path(goal, current\_position, pan, tilt, known\_obstacle\_map)$ employs the A^* algorithm to determine the shortest path to the predefined goal.

The function *simulate_camera_view*(*goal*, *known_obstacle_map*, *current_position*, *pan*, *tilt*) utilizes the `projectPoints(objectPoints, rvec, tvec, cameraMatrix, distCoeffs)` function from the Python OpenCV library [6]. Before using this function, we need to transform the 2*D* *known_obstacle_map* into a 3*D* world map for `objectPoints` as specified by the library. Depending on the approach, the world map consists of either a binary array (for static pan alignments and pan to follow path) or an array containing costs based on information about the known/unknown map, the rover's current position, or the path. Additionally, we must determine the required translation and rotation vectors (`rvec` and `tvec`) from the world coordinate origin to the rover's current position and pan-tilt alignment. The `cameraMatrix` represents the intrinsic matrix for the projection from 3*D* points in the camera coordinate system to 2*D* pixels on the image plane. The `distCoeffs` represent the distortion coefficients. In our simplified simulation, we assume the use of an ideal camera without distortion, so these coefficients are set to `None`. Subsequently, we can eliminate all projected points that are out of the camera frame's range and sum the costs from the remaining points to obtain the total *seen_points* and *seen_costs* for the particular pan sample.
In each iteration step along the path points we simulate the camera projection over the pan samples. We then choose the pan with the lowest seen costs. Finally, we update our map and current position based on this chosen pan and move on to the next step on the path.

5.2 Follow Path Pan Alignments

To calculate the best pan alignment angle, we need to compute the difference between each pan sample and the target point on the path or the goal, depending on the length of the remaining path. The implementation can be represented as follows:

```
function COST_FUNC_TO_FOLLOW_PATH(path, goal, [pan_alignmens])
    if len(path) < 10 then
        point_to_track = goal
    else
        point_to_track = path[10]
    α = calculate_angle_to_goal()
    for pan in [pan_alignmens] do
        cost_array.append(α − pan)
    return cost_array
```

5.3 Cost Function with Lateral and Longitudinal Distance to Path

The following pseudocode demonstrates the implementation of the previously discussed cost function, which takes into account the lateral distance to the path and the longitudinal distance to the rover's current position: [1]

function COST_FUNC_WITH_LATERAL_LONGITUDINAL_DISTNACE(known_map, path, current_pos)
    $lat, long = get\_information\_gain(path, current\_pos)$
    $cost\_map = 2 - exp(-0.1 \cdot long) - exp(-0.2 \cdot lat)$
    $cost\_map[known\_map] = 2$
    return *cost_map*

5.4 Cost Function with Distance to Border

The cost function incorporating distances to the boundary of the unknown terrain utilizes the `distance_transform_cdt()` method, which returns a $2D$ array cost map, from the `scipy.ndimage` Python library [15]. This function computes the distance within the input array by replacing each non-zero element with its shortest distance to any zero-valued element. Therefore, this strategy only requires knowledge of the known and unknown map, resulting in a reduction in computational complexity. In our scenario, the input array represents the map of known (1) and unknown (0) areas:

function COST_FUNC_DISTANCE_TO_BORDER(known_map)
    return $distance\_transform\_cdt(known\_map)$

6 Results and Discussion

In this chapter, we will validate and evaluate the different cost function approaches discussed during this thesis. We will analyze different scenarios using the previously mentioned metrics and varying amounts of pan alignments for better refinement. All evaluated measurements for the path exploration with a tilt angle of -30° are shown in Figure 6.4.

6.1 Collision

We first focus on demonstrating the occurrence of collisions and passing unknown since it is a hard requirement for our analysis. We observed in several scenarios a collision. The Figure [6.1c] presents scenario 5 with a collision with the static camera pan, aligned to the rover's body. This happens because of the path's sharp curve and the camera not scanning the path ahead. This demonstrates that camera movement is not only beneficial but also necessary for this task.

cost function	collision	total expl.	path length	avg time
static	True	2427	76	0
follow path	False	2161	75	8.6564e-05
long_lat	False	3704	76	5.5152e-05
border dist	False	3759	76	5.4661e-05

Table 6.1: Measurements with 5 pan samples for scenario 5 - on occurrence of collision; total map exploration in map points; path length in path points; estimated average time for cost function calculation in seconds

(a) (b) (c)

Figure 6.1: Simulation fragments for scenario 5 with static pan alignment, showing a collision in 6.1c

6.2 Total map exploration

An important metric for our experiment is the comprehensive exploration of the path. Table 6.2 presents the summarized measurements indicating the number of map points captured by the LRU as it traverses towards the goal point.

scen.	static	follow path		long_lat		border dist	
	-	5	15	5	15	5	15
0	1404	1477	1353	2401	2533	2439	2382
1	1960	1967	1872	3638	3505	3556	3524
2	2985	2701	2652	5024	4827	5157	5150
3	1941	1714	1667	2995	3066	3019	2966
4	2046	1631	1618	2785	2775	2771	2702
5	2427	2161	2028	3704	3778	3759	3738
6	1770	1630	1586	3112	3201	3109	3040
7	1801	1573	1575	2941	3060	2993	2965
8	1279	1259	1246	2354	2299	2216	2217
9	2355	2050	2019	3503	3431	3248	3351
10	1987	1546	1484	2714	2878	2543	2601
			min		max		

Table 6.2: Measurements of total map exploration in map points for each cost function method and varying pan samples (5 and 15) across scenarios 0 to 10. Minimal exploration coverage is color-coded in red, while maximal exploration coverage is color-coded in green.

From the measurements, we can conclude that the minimum exploration (red cells) is achieved by the strategy of the camera using the method to follow the path and with static pan alignment. These results are expected, given the minimal camera movement involved and show that this strategy is insufficient for the exploration task.
The maximum exploration (green cells) is achieved with the cost function that incorporates both lateral and longitudinal distances, especially by utilizing the maximum number of pan samples. This can be explained by the method considering not only the distance to the path but also the distance to the LRU's position, independently of the path. The maximum exploration is followed by the function that aligns the camera angle to the nearest unknown terrain.

Another conclusion we can make from our measurements is that the amount of samples influences the overall exploration outcome. For instance, an increase in the number of samples utilized within the path-following cost function leads to a reduction in the explored area. Since angle deviation is reduced with increased sample size the focus on the path can be calculated more precisely. Our measurements show that increasing the number of samples for the cost function considering lateral and longitudinal distance leads to a more comprehensive exploration of the rover's environment overall. The average relative difference of explored map points of is 0.94. Therefore, it would be interesting to examine in future work which sample size for pan orientation would be the most effective. Conversely, the method calculating the closest distance to the border we can observe the opposite trend of increasing total exploration with less sample angles.

6.3 Path Length

From a path length perspective, we can deduce that the path-following cost function yields the shortest traversed route across all scenarios, but difference is not significant. These measurements are presented in Table 6.4.

6.4 Average Time for Cost Map Calculation

The estimated average runtime for cost function computation is detailed in Table 6.4. Since no cost function was calculated for the static forward-facing pan alignment, no time calculation was measured. The recorded time spans a range of $\times 10^{-5}$ seconds for the cost function to follow the path ahead and range of $\times 10^{-3}$ seconds for cost functions generating a cost map, indicating that the proposed methods can be executed in real-time on the system.

In addition, we observed that the average processing time for determining the best pan angle did not increase proportionally with the number of pan samples. Even though we tripled the number of samples, the measured average processing time was 35.7% less than the expected time based on the tripled average time. This suggests that there might be an optimal sample size that offers a good balance between exploration coverage and processing speed. Further investigation is needed to determine the optimal sample size for this task.

# pan samples	follow path	long_lat	border dist
5	8.623985e-05	0.008509	0.008711
15	0.000108	0.019519	0.019474

Table 6.3: Average computational time measurements for each cost function method and varying pan samples (5 and 15) over all scenarios (0 to 10)

All collected measurements are detailed in 6.4:

Scen.	cost function	# samples	collision	total expl.	path len	avg time
0	static	-	True	1404	46	0
	follow path	5	False	1477	45	9.06505e-05
	follow path	15	False	1353	45	0.00011
	long_lat	5	False	2401	46	0.00635
	long_lat	15	False	2533	46	0.01472
	border dist	5	False	2439	46	0.00623
	border dist	15	False	2382	46	0.014515
1	static	-	True	1960	66	0
	follow path	5	False	1967	65	8.35840e-05
	follow path	15	False	1872	65	0.00010
	long_lat	5	False	3638	66	0.01023
	long_lat	15	False	3505	66	0.02219
	border dist	5	False	3556	66	0.01018
	border dist	15	False	3524	66	0.02225
2	static	-	True	2985	101	0
	follow path	5	False	2701	100	8.74769e-05
	follow path	15	False	2652	100	0.00011

Scen.	cost function	# samples	collision	total expl.	path len	avg time
	long_lat	5	False	5024	101	0.01387
	long_lat	15	False	4827	101	0.03219
	border dist	5	False	5157	101	0.01387
	border dist	15	False	5150	101	0.03236
3	static	-	True	1941	56	0
	follow path	5	False	1714	55	8.64911e-05
	follow path	15	False	1667	55	0.00011
	long_lat	5	False	2995	56	0.00782
	long_lat	15	False	3066	56	0.01821
	border dist	5	False	3019	56	0.00786
	border dist	15	False	2966	56	0.01829
4	static	-	True	2046	54	0
	follow path	5	False	1631	49	7.70920e-05
	follow path	15	False	1618	49	9.39111e-05
	long_lat	5	False	2785	51	0.00808
	long_lat	15	False	2775	50	0.01722
	border dist	5	False	2771	50	0.00787
	border dist	15	False	2702	50	0.01723
5	static	-	True	2427	76	0
	follow path	5	False	2161	75	8.58072e-05
	follow path	15	False	2028	75	0.00011
	long_lat	5	False	3704	76	0.01088
	long_lat	15	False	3778	76	0.02477
	border dist	5	False	3759	76	0.01078
	border dist	15	False	3738	76	0.02496
6	static	-	True	1770	61	0
	follow path	5	False	1630	60	8.21890e-05
	follow path	15	False	1586	60	0.00010
	long_lat	5	False	3112	61	0.01418
	long_lat	15	False	3201	61	0.02520
	border dist	5	False	3109	61	0.01357
	border dist	15	False	3040	61	0.02528
7	static	-	True	1801	55	0
	follow path	5	False	1573	54	8.32152e-05
	follow path	15	False	1575	54	0.00011
	long_lat	5	False	2941	55	0.00754
	long_lat	15	False	3060	55	0.01763

Scen.	cost function	# samples	collision	total expl.	path len	avg time
	border dist	5	False	2993	55	0.00768
	border dist	15	False	2965	55	0.01759
8	static	-	True	1279	46	0
	follow path	5	False	1259	45	8.85308e-05
	follow path	15	False	1246	45	0.00011
	long_lat	5	False	2354	46	0.00663
	long_lat	15	False	2299	46	0.01579
	border dist	5	False	2216	46	0.00672
	border dist	15	False	2217	46	0.01561
9	static	-	True	2355	72	0
	follow path	5	False	2050	70	8.68827e-05
	follow path	15	False	2019	71	0.00011
	long_lat	5	False	3503	71	0.01015
	long_lat	15	False	3431	69	0.02319
	border dist	5	False	3248	71	0.01012
	border dist	15	False	3351	71	0.02364
10	static	-	True	1987	61	0
	follow path	5	False	1546	52	8.49827e-05
	follow path	15	False	1484	52	0.00011
	long_lat	5	False	2714	53	0.00739
	long_lat	15	False	2878	55	0.01758
	border dist	5	False	2543	53	0.00825
	border dist	15	False	2601	53	0.01709

Table 6.4: Measurements for scenarios 0 to 10 include the following: occurrence of collision (highlighted in red if true), total map exploration in map points (with maximal exploration marked in green), path length in path points, and estimated average time for cost function calculation in seconds. s

6.5 Smooth Camera Dynamic

Thus far, we have not accounted for the dynamics of camera movement, operating under the assumption that the camera moves significantly faster than the LRU. This aspect must be addressed in future development. Large deviations in pan angles during each iteration can negatively affect the localization process due to abrupt shifts from one an-

gle to another. This issue is particularly dominant in the cost function that calculates the closest distance to the unknown area. Therefore, we introduced an additional method that considers camera movement, ensuring smoother angular velocity. This method can be integrated into any cost function by incorporating an additional weighting factor for pan angle deviation relative to the previous iteration step. In Figure 6.2, we show the camera movement within a segment of scenario 0, comparing the behavior with integrated dynamic weighting (on the right) to that without dynamic weighting (on the left).

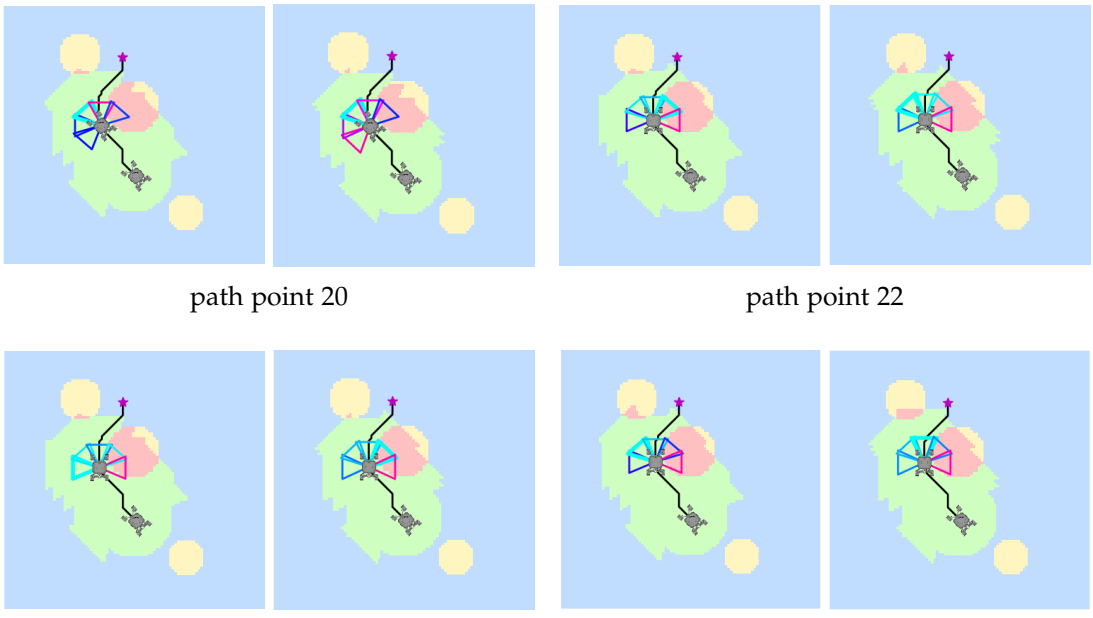


Figure 6.2: Comparison of Scenario 0 with (right) and without (left) dynamic weighting; a detailed evaluation is provided in 6.5

We observe that the pan amplitude is greater without dynamic weighting compared to the same cost function strategy with dynamic weighting. The pan angles for the version without dynamic weighting vary from $0°$ to $-90°$ to $45°$ to $-45°$, covering a total of $315°$, while the pan angles for the dynamic function vary from $0°$ to $45°$ to $-45°$ to $0°$, covering a total of $180°$.
The mathematical formulation can be described as follows:

$$\min_{\beta \in pan\_samples} = |(prev\_pan - \beta) \cdot \lambda| + cost(\beta) \tag{6.1}$$

where $cost(\beta)$ is the total cost value for the pan sample, *prev_pan* is the chosen pan angle from the previous iteration step and λ is the weight which must be adjusted for each cost function.

Form the measurements in Table 6.5 we can conclude that a smoother camera dynamic lead to less comprehensive exploration along the path. However, in some cases, the total exploration with smoother camera dynamics is actually greater. This can be attributed to the different paths traversed; the order in which the camera pans affects the observed environment, leading to a different traversal path and consequently changing the total exploration of the environment. Achieving the required camera smoothness can be further refined through parameter tuning of the weighting, which is outside the scope of this bachelor's thesis.

Scenario	# samples	total expl. w/ weighting	total expl. w/o weighting
0	5	2387	2439
0	15	2395	2382
1	5	3530	3556
1	15	3444	3524
2	5	4845	5157
2	15	4969	5150
3	5	2952	3019
3	15	2922	2966
4	5	2659	2771
4	15	2855	2702
5	5	3674	3759
5	15	3633	3738
6	5	2999	3109
6	15	3118	3040
7	5	2845	2993
7	15	2848	2965

Table 6.5: Measurements for total map exploration in map points with distance to unknown border cost function w/ and w/o dynamic weighting for Scenarios 0 to 7; larger exploration is marked as green, while less exploration is marked red

6.6 Main Results

In this thesis, our goal was to implement a cost function for the path-following task of the LRU to expand its environmental exploration capabilities. We proposed various strategies and compared them using multiple metrics across different scenarios and varying numbers of pan samples. The contributions of this thesis consist of the following main findings:

First, we concluded that a static camera pan-tilt alignment and the path-following method are insufficient for this task. The former approach violates hard requirements and led to collisions or traversed through unknown areas in some testing scenarios, indicating that this strategy must be discarded. Similarly, the path following method produced the poorest results in the exploration task.

Second, from the perspective of total exploration, the cost function inspired by Boyu Zho et al. [1] detected most of the surroundings along the path independently of the sample size of pan alignments.

Third, we examined a cost function that calculates the distance to unknown terrain and prioritizes the closest areas. This approach satisfied all hard requirements, preventing collisions and traversal through unknown areas. A benefit of this strategy is that it relies only on the information from the known map for its computation and does not require any knowledge about the path, which would increases the computational complexity.

In summary, we conclude that approach which incorporates the lateral distance to the path and the longitudinal distance relative to the rover's current position, based on Boyu Zho et al. [1] is the most suiting strategy for expanding the environment in the path following task.

7 Conclusion and Further Work

In this thesis, we implement a cost function for the path-following task of the LRU to expand its environmental exploration capabilities. We proposed various strategies and compared them using multiple metrics across different scenarios and varying numbers of pan samples. We showed that the proposed cost function based on previous work of Boyu Zho et al. [1] which incorporates lateral distance to the path and longitudinal distance to the rover's position is the best suiting strategy to achieve our goals.

From our measurements, we can derive that the runtime for the cost function computation is fast enough to be executed in real-time on the system. Therefore, it is interesting to test the implemented cost functions on the LRU in further development.

During the experiment, we noticed that both methods incorporating cost functions, show a hectic panning of the camera at each iteration step, which cause significant issues in the localization task. Therefore, it is crucial to focus on the dynamics of the camera to ensure smoother pan movements between iterations. To address this issue, we introduced a method that additionally weights each pan sample after the cost map evaluation. However, it is interesting to continue developing this approach in future work and examine the resulting behavior for each cost function. The foundational implementation is provided, but it requires further parameter tuning for each cost function.

Another observation we made is that the sample size for the pan alignments affects the camera's exploration behavior, it is interesting to investigate in further development the best effective number of pan alignments so that the exploration is extended while the time for computing the cost function is minimized.

List of Figures

List of Tables

Bibliography

[1] B. Z. et al. “RAPTOR: Robust and Perception-Aware Trajectory Replanning for Quadrotor Fast Flight.” In: *IEEE Transactions on Robotics* 37.6 (2021).

[2] S. Bai, J. Wang, F. Chen, and B. Englot. “Information-theoretic exploration with Bayesian optimization.” In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 1816–1822.

[3] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart. “Receding horizon" next-best-view" planner for 3d exploration.” In: *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2016, pp. 1462–1468.

[4] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart. “Receding horizon" next-best-view" planner for 3d exploration.” In: *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2016, pp. 1462–1468.

[5] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. F. Durrant-Whyte. “Information based adaptive robotic exploration.” In: *IEEE/RSJ international conference on intelligent robots and systems*. Vol. 1. IEEE. 2002, pp. 540–545.

[6] G. Bradski. “The OpenCV Library.” In: *Dr. Dobb's Journal of Software Tools* (2000).

[7] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O’Reilly Media, Inc.", 2008.

[8] D. Cremers. *Computer Vision II: Multiple View Geometry*. May 2024.

[9] DLR. *LRU*. 2024. URL: https://www.dlr.de/de/rm/forschung/robotersysteme/mobile-plattformen/lru.

[10] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Ed. by R. Hartley. Cambridge Univ. Press, 2003.

[11] M. Kuhne, R. Giubilato, D. Leidner, and M. A. Roa Garzon. “Addressing the Entry Barrier for Experimentation in Perception-aware Trajectory Planning for Planetary Rovers.” In: *IEEE International Conference on Robotics and Automation (ICRA) 2023 Workshops*. 2023.

[12] H. Li. *Computer Vision II: Multiple View Geometry*. May 2023.

[13] N. Sturtevant. "Benchmarks for Grid-Based Pathfinding." In: *Transactions on Computational Intelligence and AI in Games* 4.2 (2012), pp. 144–148.

[14] M. Vayugundla, M. Kuhne, A. Wedler, and R. Triebel. "Datasets and Benchmarking of a path planning pipeline for planetary rovers." In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Workshop: Evaluating Motion Planning Performance*. Oct. 2022.

[15] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[16] B. Yamauchi. "A frontier-based approach for autonomous exploration." In: *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*. IEEE. 1997, pp. 146–151.

[17] M. Yi, S. Soato, J. Košecká, and S. Sastry. *An Invitation to 3-D Vision*. Ed. by S. S. Antman. Springer New York, 2004.