Space Robotic Systems - Mars Rover

Leonardo Russo

February 10, 2024

Abstract

This report delves into the advanced navigational and localization techniques employed in the Mars Rover project within the challenging environment of the Gale Crater on Mars. The primary focus is on three crucial tasks: navigation, path planning, and localization. In the navigation task, a sophisticated Moving to a Pose control law is developed, involving coordinate transformation and kinematic modeling to precisely guide the rover to desired poses. For path planning, the efficient A* algorithm is implemented, facilitating optimal pathfinding through obstacle-laden terrain. The final task, localization, is addressed using the Extended Kalman Filter (EKF), a robust method for estimating the rover's state in dynamic conditions. This report not only outlines the theoretical foundations of these techniques but also presents their practical application in the Mars Rover's journey, ensuring its successful exploration and data collection on the Martian surface.

Contents

1 Introduction **2**

2 Navigation **2**
2.1 Coordinate Transformation 2
2.2 Control Strategy 2
2.3 Kinematic Model 2

3 Path Planning **6**
3.1 A* Algorithm 6

4 Localization **8**
4.1 Dead Reckoning 8
4.2 Extended Kalman Filter 9

https://github.com/Leonardo-Russo/Autonomous-Driving-Rover

1 Introduction

NASA's Mars Curiosity Rover, an advanced mobile robot, is currently conducting a vital exploration mission within the Gale Crater on Mars. This mission primarily focuses on investigating the Martian surface to identify past or present conditions that could potentially support life. The rover's design and operational capabilities are central to its success in this challenging task.
Some of the specifications of the Curiosity Rover that are relevant to our study include:

- V_{max}**:** the rover is capable of reaching a maximum velocity of $4\,cm/s$, which is suitable for the cautious exploration of the Martian terrain.
- L**:** the distance between the axles of the rover, also known as the wheel base, is approximately 3 meters.

2 Navigation

This section delves into the development of a sophisticated Moving to a Pose Control law, which is pivotal for enabling the Curiosity Rover to precisely attain a desired final pose on the Martian surface.
The initial and final poses are provided as

$$\underline{P_0} = \begin{bmatrix} 42.38\,km & 11.59\,km & \frac{\pi}{2} \end{bmatrix} \tag{2.1}$$

$$\underline{P_1} = \begin{bmatrix} 33.07\,km & 19.01\,km & \pi \end{bmatrix} \tag{2.2}$$

2.1 Coordinate Transformation

Unlike the conventional approach, my implementation necessitates a slightly modified coordinate transformation $(x, y, \theta) \to (\rho, \alpha, \beta)$ since the final desired pose is not null.
In particular, the forward and backwards transformations are provided below,

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \to \begin{bmatrix} \rho \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \sqrt{(x-x^*)^2 + (y-y^*)^2} \\ \arctan 2(y-y^*, x-x^*) - \theta \\ -\alpha - \theta + \theta^* \end{bmatrix} \tag{2.3}$$

$$\begin{bmatrix} \rho \\ \alpha \\ \beta \end{bmatrix} \to \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} -\rho\cos(\alpha+\theta) + x^* \\ -\rho\sin(\alpha+\theta) + y^* \\ -\alpha - \beta + \theta^* \end{bmatrix} \tag{2.4}$$

where (x^*, y^*, θ^*) represent the coordinates and orientation of the reference pose.

2.2 Control Strategy

In order to implement a kinematic model, we must first introduce the control variables as the linear and angular velocities (v, ω).
These will be found by means of three proportional gains as,

$$v = K_\rho \rho \tag{2.5}$$

$$\omega = K_\alpha \alpha + K_\beta \beta \tag{2.6}$$

where K_ρ, K_α, and K_β are the proportional gains for the trajectory optimization, ensuring effective obstacle avoidance.

2.3 Kinematic Model

Finally, the cornerstone of the navigation strategy is the kinematic model, which approximates the propagation of the rover's reference state over time with respect to a more complex dynamic model. Now, with

the establishment of a control law, we can finally delineate the evolution of the rover's relative state using the following differential equations

$$\begin{bmatrix} \dot{\rho} \\ \alpha \\ \beta \end{bmatrix} = \begin{cases} -v\cos(\alpha) \\ \frac{v\sin(\alpha)}{\rho} - \omega \\ -\frac{v\sin(\alpha)}{\rho} \end{cases} \tag{2.7}$$

This formulation sets the stage for implementing an Euler integration approach to compute the rover's evolving state.
A meticulous process of fine-tuning the control gains was undertaken to ensure a navigation trajectory that successfully avoids obstacles. The trajectory and the corresponding control gains employed in this navigation task are presented below.

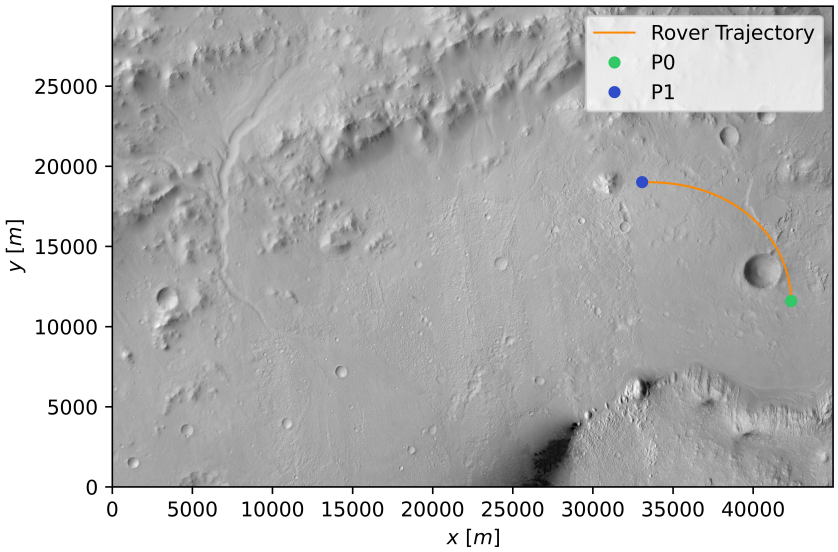

Figure 1: Navigation Trajectory on the Map

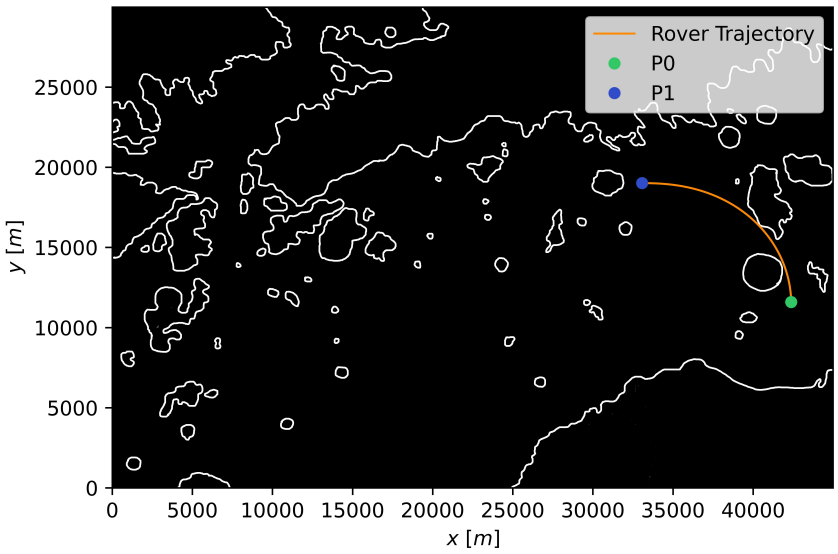

Figure 2: Navigation Trajectory on the Obstacle Map

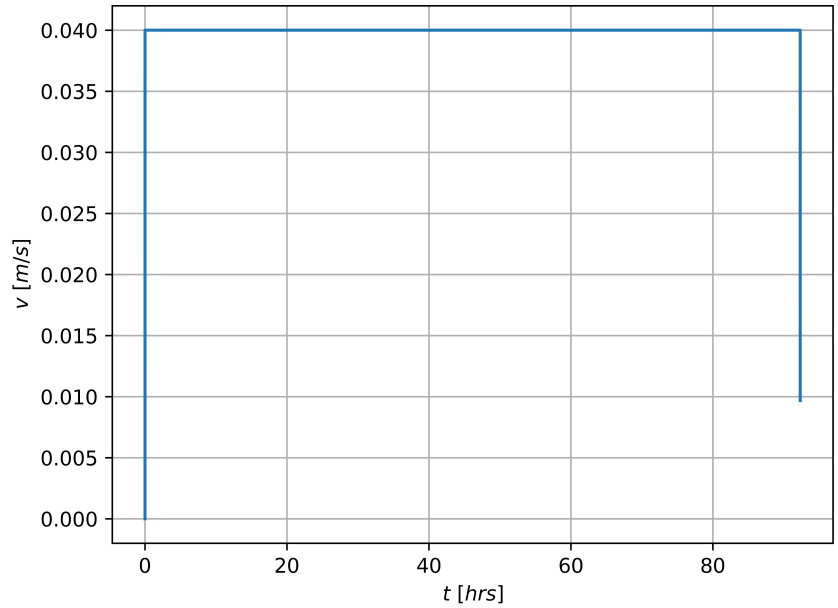

Figure 3: Velocity during the Trajectory

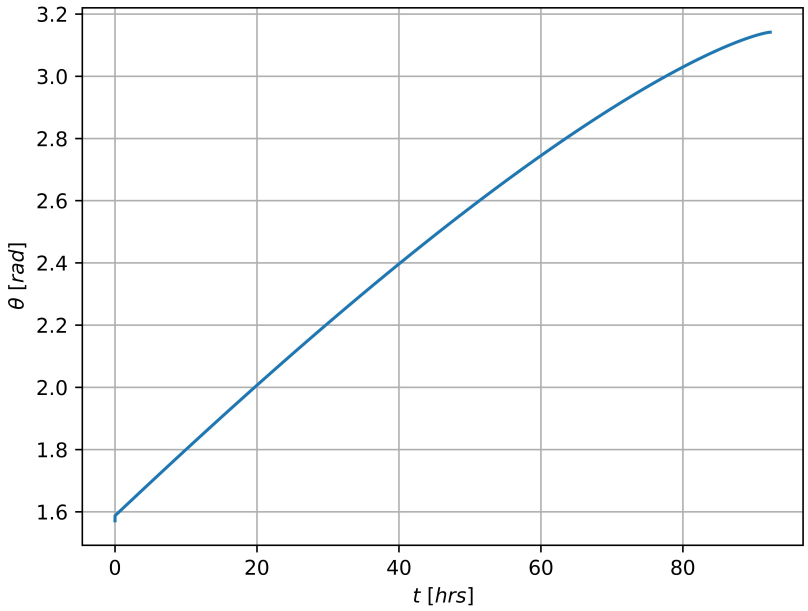

Figure 4: Heading Angle during the Trajectory

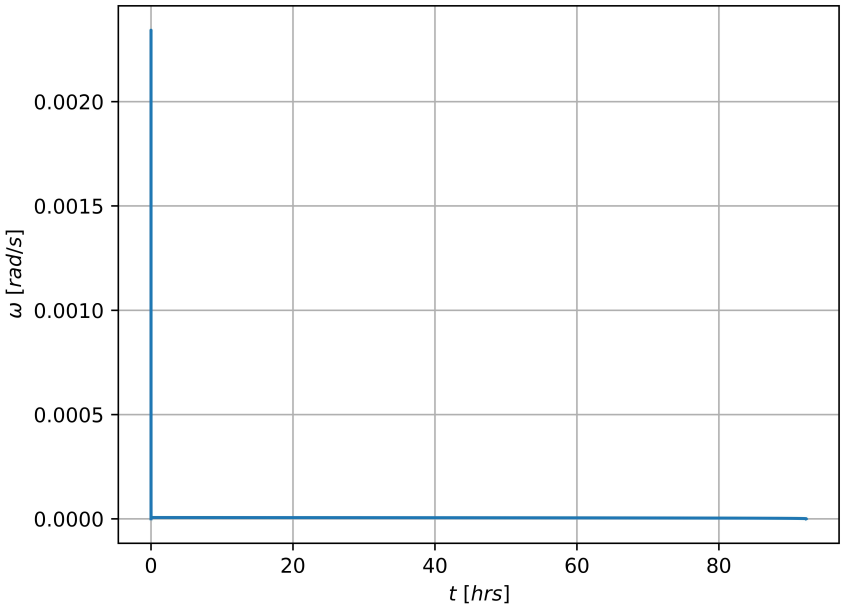

Figure 5: Heading Angular Velocity during the Trajectory

Gain	Value
K_ρ	0.100
K_α	0.145
K_β	-0.190

Table 1: Fine-Tuned Control Gains

Time Required: $3.85\,days$

Final State Error: $\begin{bmatrix} 8.71 \cdot 10^{-2}\,m & 2.21 \cdot 10^{-9}\,m & 8.47 \cdot 10^{-7}\,rad \end{bmatrix}$

To ensure the validity of our results, especially given the proximity of the rover's trajectory to potential obstacles, a numerical check was performed. This analysis confirmed that the trajectory successfully avoids intersecting with any obstacles, even though the margins might appear narrow visually.

3 Path Planning

In this section of the work, the Rover is required to solve the pathfinding problem in order to find a reference trajectory which, starting from the final pose of the Navigation Task and avoiding the obstacles provided in the obstacle map, leads to pose P_2 defined as

$$\underline{P_2} = \begin{bmatrix} 10.87 & 25.67 \end{bmatrix} \qquad km \tag{3.1}$$

It is important to note that the pathfinding problem is best tackled in the pixel domain, since the conversion from pixels to real world coordinates is not injective.

3.1 A* Algorithm

The A* algorithm is a highly efficient pathfinding and graph traversal method, blending the best aspects of Dijkstra's Algorithm and Greedy Best-First-Search. In the Mars Rover project, A* is crucial for navigating the rover through complex terrains with obstacles, calculating an optimal path to the target. Due to the complexity of A*'s operational steps, a flowchart is presented below for a clear and concise visual understanding of the process.

Note that the implementation of the algorithm followed in this work is not completely adherent with the algorithm provided during the course. Namely, among other differences, most of which regarded the different programming languages used, one can notice the absence of the camefromSet. The reason behind this is that, by leveraging python's ability to create hashable classes, it was easier to include a parent attribute to each node rather than utilizing lists. This means that, as the nodes get explored, the history of the exploration is stored inside each current node as a nested class attribute, which then can be retrieved with a recurring algorithm.

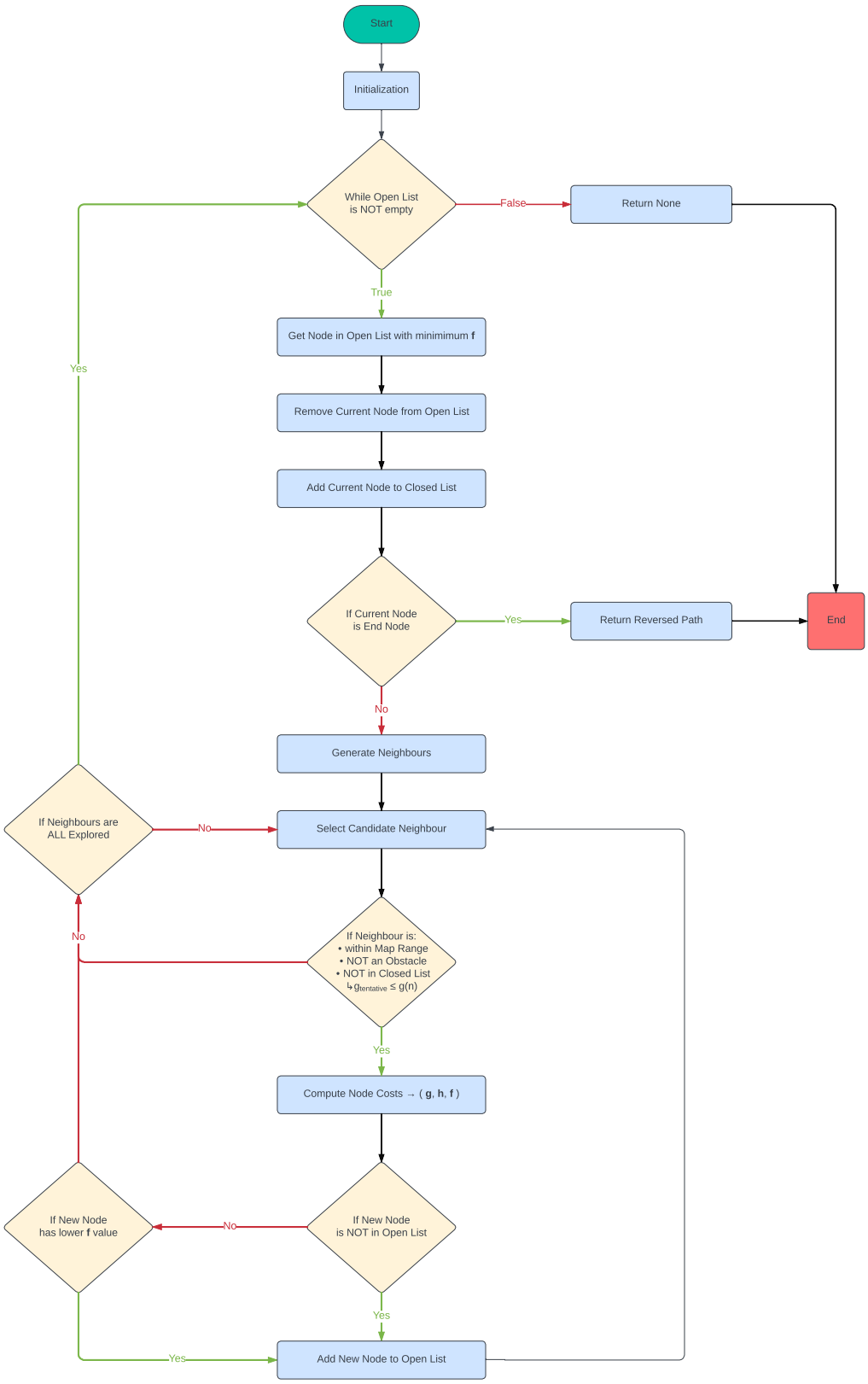

Figure 6: Flowchart of the A* Algorithm

Therefore, by exploiting the algorithm above, I was able to retrieve the following path,

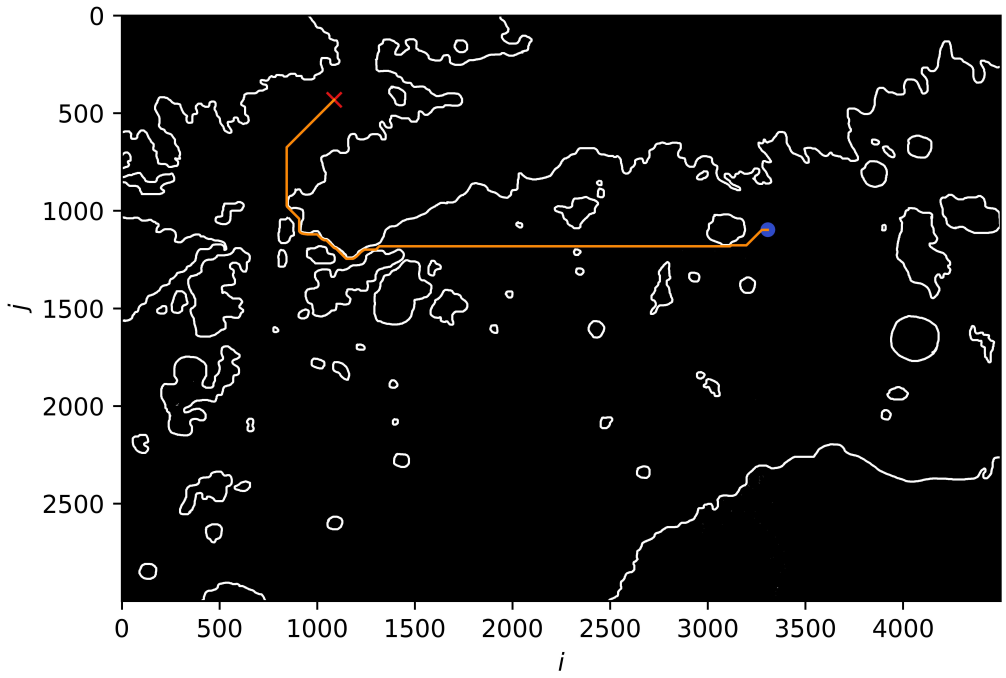

Figure 7: Reference Path from A*

4 Localization

The Extended Kalman Filter (EKF) is integral to our Mars Rover's localization, providing a robust method for estimating the rover's state in a dynamic Martian environment. Leveraging the EKF, we enhance the rover's ability to accurately navigate and understand its position, crucial for mission success.

4.1 Dead Reckoning

The Trajectory Reconstruction begins by predicting the rover's next state based on its current state, motion, and inherent uncertainties. This prediction step, often referred to as dead reckoning, is crucial when direct measurements are unavailable or sparse. The state's evolution, considering the rover's movement and noise in the odometry data, is described by

$$\begin{cases} \bar{x}_{k+1} & = \hat{x}_k + (\delta_d + \nu_d)\cos(\hat{\theta}_k), \\ \bar{y}_{k+1} & = \hat{y}_k + (\delta_d + \nu_d)\sin(\hat{\theta}_k), \\ \bar{\theta}_{k+1} & = \hat{\theta}_k + \delta_\theta + \nu_\theta \end{cases} \tag{4.1}$$

Simultaneously, we update the covariance matrix P to reflect the uncertainty in the state estimate after this motion,

$$\bar{P}_{k+1} = F_q \hat{P}_k F_q^T + F_v \hat{V} F_v^T \tag{4.2}$$

where

$$F_q = \left.\frac{\partial f}{\partial q}\right|_{v=0} = \begin{bmatrix} 1 & 0 & -\delta_d \sin\theta \\ 0 & 1 & \delta_d \cos\theta \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

$$F_v = \left.\frac{\partial f}{\partial v}\right|_{q=0} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \tag{4.4}$$

4.2 Extended Kalman Filter

Upon acquiring new observations, such as from LIDAR, the EKF updates the estimated state. The observation model $h(q, x_{\text{lm}}, y_{\text{lm}})$ correlates the actual state with sensor readings,

$$\begin{cases} r = \sqrt{(x_{\text{lm}} - x)^2 + (y_{\text{lm}} - y)^2}, \\ \beta = \text{atan2}(y_{\text{lm}} - y, x_{\text{lm}} - x) - \theta \end{cases} \tag{4.5}$$

where r is the range and β is the bearing angle.

The Kalman Gain K is then computed to optimally merge the prediction with the new measurements,

$$K = \bar{P} H_q^T (H_q \bar{P} H_q^T + H_w W H_w^T)^{-1} \tag{4.6}$$

where

$$H_q = \begin{bmatrix} \frac{x - x_{lm}}{\sqrt{(x-x_{lm})^2 + (y-y_{lm})^2}} & \frac{y - y_{lm}}{\sqrt{(x-x_{lm})^2 + (y-y_{lm})^2}} & 0 \\ -\frac{y - y_{lm}}{(x-x_{lm})^2 + (y-y_{lm})^2} & \frac{x - x_{lm}}{(x-x_{lm})^2 + (y-y_{lm})^2} & -1 \end{bmatrix} \tag{4.7}$$

and

$$H_w = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{4.8}$$

Finally, we update the rover's state and covariance matrix, assimilating new information and reducing uncertainty,

$$\hat{q}_{k+1} = \bar{q}_{k+1} + K_{k+1}[z_{k+1} - h(\bar{q}_{k+1})] \tag{4.9}$$

$$\hat{P}_{k+1} = (I - K H_q) \bar{P}_{k+1} \tag{4.10}$$

Following this approach, it is possible to perform the trajectory reconstruction yielding the following results, where the uncertainty ellipses are shown with a confidence of 3σ, hence ensuring a 99.73% probability that the true state lies within the ellipse.

As a final remark the author clarifies that although the sizes of the uncertainty ellipses can appear mildly incosistent on the graphs, upon a closer inspection it is clear that this is only an issue with the graphical representation provided by the `matplotlib` library. This can be proved by reducing the resolution of the image or simply by looking at the determinant of the covariance matrix which has the expected sawtooth behaviour.

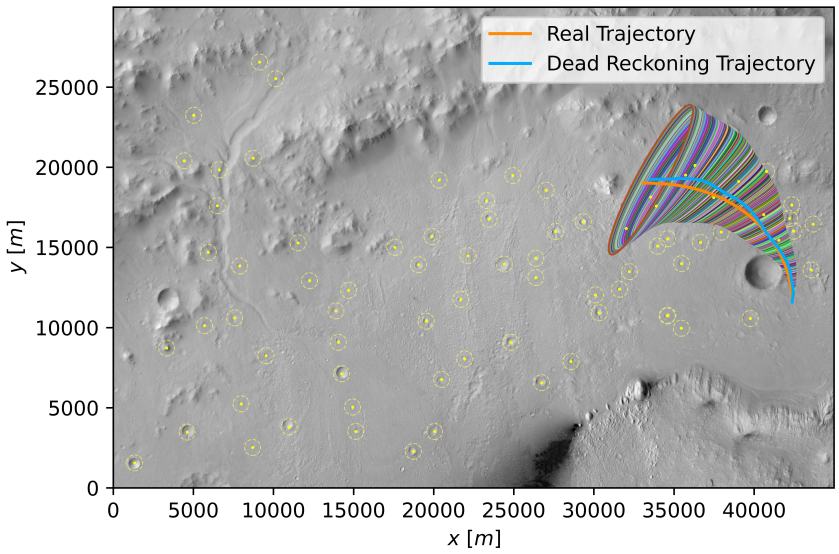

Figure 8: Reconstructed Trajectory using only Dead Reckoning

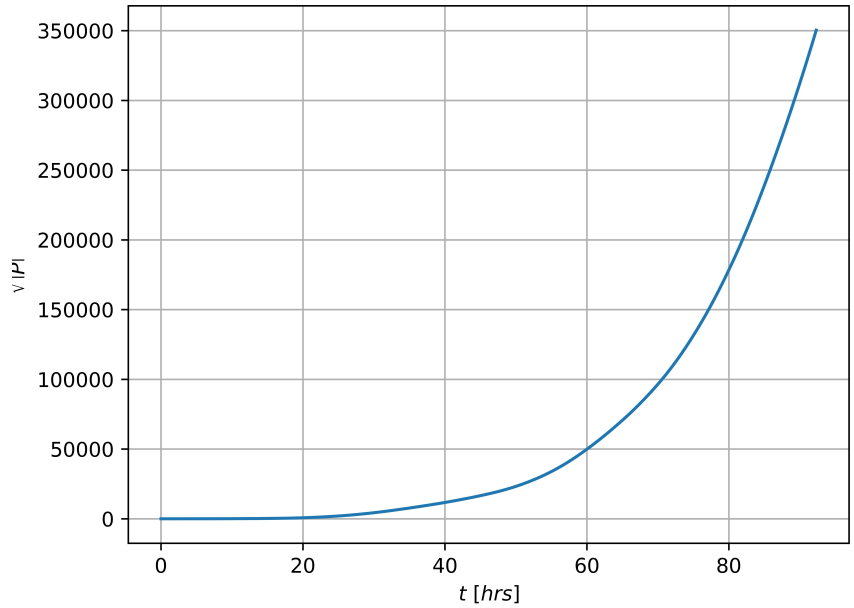

Figure 9: Covariance Matrix during Dead Reckoning

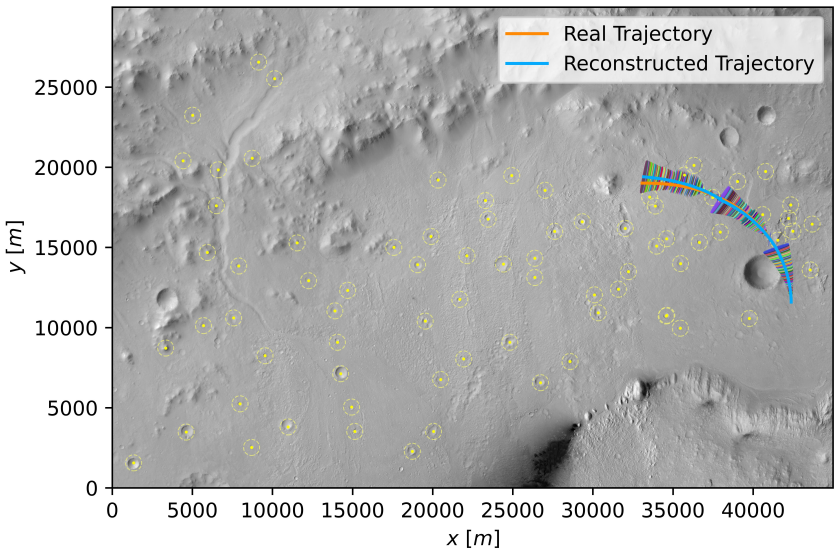

Figure 10: Reconstructed Trajectory using EKF

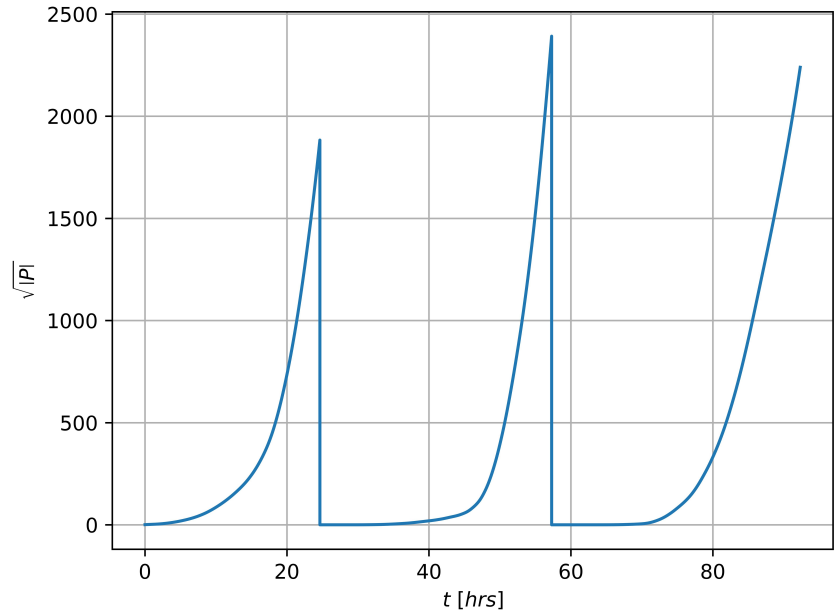

Figure 11: Covariance Matrix during EKF