INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA DE INTELIGÊNCIA ARTIFICIAL




Multi-Agent Exploration of Unknown Terrain: Combining Neural Time Series Forecasting with Multi-Agent Coordination for Planetary Surface Exploration

IVAN MANUEL ESTEVÃO VIEGAS
Outubro de 2025

P.PORTO

Multi-Agent Exploration of Unknown Terrain

Combining Neural Time Series Forecasting with Multi-Agent Coordination for Planetary Surface Exploration

Ivan Manuel Estevão Viegas

Student No.: 1222735

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science, Specialisation Area of Artificial Intelligence

Supervisor: Luís Conceição

Evaluation Committee:
President:
Luís Filipe de Oliveira Gomes, Assistant Professor, Institute of Engineering, Polytechnic of Porto

Members:
Manuel Rodrigues, Assistant Researcher, School of Engineering, University of Minho
Luís Manuel Silva Conceição, Assistant Professor, Institute of Engineering, Polytechnic of Porto

Porto, September 30, 2025

Abstract

This work addresses the challenge of autonomous robotic exploration in uncertain planetary environments, such as the Martian surface, where long communication delays with Earth make local decision-making essential. The main objective was to develop a simulation framework aimed at enhancing the efficiency and autonomy of multi-agent robotic systems, with a particular emphasis on scalable cooperation. The study investigates how the integration of Multi-Agent Reinforcement Learning (MARL) algorithms and dynamic environmental simulations can improve exploration in unknown terrains.

The methodology involved the development of a detailed 2D simulation environment of Jezero Crater, Mars, incorporating realistic weather conditions modeled from the MEDA dataset. Climate forecasting relied on N-BEATS and LSTM models, with N-BEATS demonstrating superior performance in predicting environmental variables. Coordination between Explorer agents (tasked with mapping and sample identification) and Transporter agents (responsible for collection and delivery) was managed through an Auction-Based Coordination and Autonomy Layer powered by MARL.

The results demonstrate the effectiveness of the multi-agent approach, achieving a successful coordination rate of 79.4% across multiple simulation runs under varying atmospheric conditions (with successful coordination defined as task assignments completed within the mission time budget via the auction-based mechanism). Furthermore, emergent behavioral specialization was observed among Transporter agents, with significant diversity in the *confidence_bias* parameter (coefficient of variation, $CV = 0.256$), enhancing overall system robustness. Additionally, the system exhibited linear scalability and a 192.9% improvement in territorial coverage in multi-agent configurations compared to single-agent setups.

In conclusion, the integration of MARL with dynamic environmental simulations provides a robust and adaptable solution to the complex challenges of robotic exploration and navigation in extreme environments, paving the way for future autonomous space missions.

Keywords: Multi-Agent Reinforcement Learning, Autonomous Robotic Exploration, Surface of Mars, Multi-Agent Coordination, Time Series Forecasting, Dynamic Environmental Simulation

Acknowledgement

I would like to express my sincere gratitude to everyone who, directly or indirectly, contributed to the completion of this thesis.

I want to thank all my family members. In particular, I am deeply thankful to my grandparents, who raised me with love and dedication, instilling in me fundamental values and giving me the strength to pursue my goals. They taught me the importance of hard work, perseverance, and never giving up on my dreams.

To my girlfriend, for her daily companionship, her unconditional support even in the most difficult moments, and the constant encouragement she gave me. To my closest friends, for their motivation and encouragement, even when I doubted I would be able to finish this work.

Finally, I would also like to acknowledge my supervisor for his guidance and support throughout the development of this thesis.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

2D	Two-Dimensional.
ACO	Ant Colony Optimization.
AI	Artificial Intelligence.
AIA	Artifical Intelligence Act.
ATS	Air Temperature Sensor.
CFPA-QL	Flower Pollination Algorithm-Q-Learning.
CMD-QL	Manhattan Distance-Q-Learning.
CNN	Convolutional Neural Network.
CTDE	Centralized Training with Decentralized Execution.
Dist-QTRL	Distributed Quantum-Train-Based Reinforcement Learning.
DL	Deep Learning.
eQMARL	Entangled Quantum Multi-Agent Reinforcement Learning.
GAN	Generative Adversarial Network.
GPU	Graphics Processing Unit.
H2GNN	Hierarchical-Hops Graph Neural Networks.
HMA-SAR	Heterogeneous Multi-Agent Search and Rescue.
HS	Relative Humidity Sensor.
HTN	Hierarchical Task Network.
ICU	Instrument Control Unit.
IQL	Independent Q-Learning.
KNN	K-Nearest Neighbors.
LIDAR	Light Detection and Ranging.
LS	Solar Longitude.
LSTM	Long Short-Term Memory.
LTST	Local True Solar Time.
MAAC	Multi-Agent Actor-Critic.
MAE	Mean Absolute Error.

MAMECS	Multi-head Attention-based Multi-robot Exploration in Continuous Space.
MARL	Multi-Agent Reinforcement Learning.
MDP	Markov Decision Process.
MEDA	Mars Environmental Dynamics Analyzer.
MER	Mars Exploration Rover.
ML	Machine Learning.
MPE	Multi-Agent Particle Environment.
MSL	Mars Science Laboratory.
POMDP	Partially Observable Markov Decision Process.
PRISMA	Preferred Reporting Items for Systematic Reviews and Meta-Analyses.
PS	Pressure Sensor.
QMARL	Quantum Multi-Agent Reinforcement Learning.
QNN	Quantum Neural Network.
QRL	Quantum Reinforcement Learning.
RAMP	Rover Avionics Mounting Panel.
RDS	Radiation and Dust Sensor.
RL	Reinforcement Learning.
RL-QMLAS	Deep Reinforcement Learning with Adaptive Search of Learning Targets.
RNN	Recurrent Neural Network.
RSM	Remote Sensing Mast.
SEG	Semantic Epsilon Greedy.
SMAC	StarCraft Multi-Agent Challenge.
SOL	Solar Day.
SOMARL	Symbolic Selection for Multi-Agent Reinforcement Learning.
TIRS	Thermal Infrared Sensor.
UV	Ultraviolet.
VAE	Variational Autoencoder.
WS	Wind Sensor.

Chapter 1

Introduction

This chapter outlines the context of this study, emphasizing the significance of the proposed recommender systems. It will elaborate on the research focus, key questions, and the methodology that will be employed.

1.1 Context

Space exploration missions face formidable challenges when navigating and operating in the uncertain and high-risk environments of planetary surfaces, particularly Mars. To address these challenges, researchers have explored the use of MARL systems, which enable autonomous agents to learn effective coordination strategies through trial-and-error interactions.(Colby, Yliniemi, and Tumer 2016; Yliniemi, Agogino, and Tumer 2014)

Actually, the advancements in autonomous technologies used by the Perseverance rover during the Mars 2020 mission have significantly improved planetary exploration efforts. Systems like AutoNav and Terrain Relative Navigation allow the rover to make real-time decisions, navigate unpredictable terrain (including the Jezero Crater delta), and avoid obstacles autonomously. However, despite these advances, challenges remain in adapting quickly to sudden environmental changes and in handling complex obstacle scenarios that can compromise rover operation Verma et al. 2023.

These advancements demonstrated by Perseverance's onboard autonomy illustrate how local decision-making can reduce routine operator load and improve responsiveness in unstructured terrain (Verma et al. 2023). However, single rovers remain limited in coverage, redundancy and task parallelism. Multi-agent systems address these limitations by distributing perception and decision tasks across cooperating units, increasing exploration throughput and fault tolerance. Such distributed autonomy is particularly relevant for missions to distant bodies (e.g., Europa or Titan), where long communication delays make continuous human oversight impractical and mission success depends on agents' local adaptability.

1.2 Problem Statement

In the current space exploration landscape, the key objective is to develop autonomous multi-agent systems able to navigate and operate in complex, uncertain environments such as the Martian surface. These systems should reduce reliance on real-time guidance from Earth-based controllers. This is necessary because the communication delay between Earth and Mars, which can be as long as several minutes due to the large distance between the two planets, makes it challenging for Earth-based operators to provide immediate, direct control

of the robotic exploration systems. As a result, it is necessary to create highly autonomous robots that can make decisions and carry out tasks without constant supervision from Earth. (Yliniemi, Agogino, and Tumer 2014).

Achieving full operational autonomy requires significant computational power to process visual data and assess potential risks. While the Perseverance rover successfully navigated more than 94% of the Rapid Traverse Campaign's routes autonomously, it still depends on Earth-based operators for high-level decision-making, such as defining scientific priorities (Verma et al. 2023). This reliance on human intervention highlights a key limitation: current autonomous systems struggle to balance computational cost and autonomous decision-making in complex environments. Moreover, scalability challenges further hinder the deployment of more sophisticated multi-agent exploration systems. Addressing these challenges requires the development of more robust and adaptive autonomous frameworks capable of operating with minimal external input (Verma et al. 2023).

The current exploration platforms demand substantial computational capacity to process high-quality visual data. These challenges underline the imperative to devise more efficient and scalable systems.

Also, research has shown that MARL systems can outperform single-agent approaches in exploring non-stationary environments, and provide a comprehensive overview of the most relevant applications in this domain (Ning and L. Xie 2024).

Concretely, the proposed solution decomposes the single, monolithic rover into a set of specialized cooperative agents, each designed for a restricted set of functions (for example: scouts for rapid mapping and obstacle avoidance; geologist agents for local sampling and in-situ analysis; transporter agents for collection and delivery; and a light-weight coordinator for mission-level task allocation).

For tractability, this project will focus on two of these classes:

- **Explorer agents**: responsible for mapping, local perception, and sample identification.
- **Transporter agents**: responsible for collection, consolidation, and delivery of samples, and examine their cooperative behaviours.

1.3 Objectives

The primary objective of this study is to develop a simulation framework that integrates MARL algorithms to enhance the efficiency or autonomy of robotic systems exploring unknown environments, such as the Martian surface, with an emphasis and scalable multi-agent cooperation.

The specific objectives are as follows:

1. Train agents in environments with systematically varied atmospheric conditions (e.g., wind speed, pressure, temperature) to evaluate robustness across scenarios.
2. Develop a system where autonomous explorers map the terrain, identify obstacles, and locate valuable scientific samples.
3. Utilize a simulated 2D range-bearing sensor (LIDAR-like) with occupancy-grid mapping and navigation algorithms to build detailed maps and detect target samples.

4. Implement MARL to enable effective cooperation and coordination among multiple rovers, improving exploration efficiency and adaptability.

The research focuses on answering the following guiding question:

How does integrating MARL algorithms with dynamic environmental simulations affect the efficiency and autonomy of robotic exploration in unknown terrains (e.g., Mars)?

By addressing this question, the study aims to bridge theoretical advancements with practical applications, contributing to the growing body of research on autonomous exploration systems. The proposed framework focuses on classical and multi-agent reinforcement learning techniques; quantum-enhanced methods (QMARL) are reviewed as a promising future direction but their implementation is out of scope for this thesis and therefore considered as follow-up or bonus work.

1.4 Contributions

This study aims to make significant contributions to the scientific understanding of autonomous robotic systems for the exploration of unknown environments, with a focus on space exploration. The implemented simulation will combine classical techniques such as RL and route optimization algorithms to evaluate cooperative behaviours and task allocation in multi-agent teams. While quantum-enhanced methods are discussed as promising future directions, they are not part of the implemented work and are treated as follow-up material.

The simulation framework will allow us to gain valuable insights into the performance of the proposed solution in the context of the Martian surface and assess the advantages of the proposed multi-agent approach. The key contributions of this study will be:

1. The design and implementation of a comprehensive 2D simulation environment representing the Martian surface, with realistic terrain, environmental conditions, and the presence of obstacles. This simulation will serve as a controlled testbed to prototype algorithms and perform systematic evaluations of agent behaviours.
2. The design and simulation-based evaluation of cooperative multi-agent policies for explorer and transporter roles. Agents will be trained and tested within the simulated environment to assess mapping throughput, sample retrieval efficiency, and resilience to sensor noise and partial observability; the work focuses on simulation experiments and does not claim deployed robotic implementations.
3. An analysis of quantum-enhanced MARL approaches QMARL as a potential avenue to extend the present work in future studies; the present thesis does not implement QMARL but discusses how such methods could complement the simulated MARL framework.

The outcomes of this study will contribute to the scientific community's understanding of multi-agent systems for solving complex exploration and navigation challenges in unknown and harsh environments, such as the Martian surface. The developed simulation and the insights gained from this research will pave the way for future work that may include quantum-enhanced techniques as an optional extension.

Chapter 2

State of the Art

For clarity and coherence, this chapter is structured into two main subsections, each addressing a distinct but complementary aspect of the study.

The first section provides a theoretical overview, summarizing the current state of research on classical MARL approaches to exploring unknown environments and analyzing emerging work that integrates concepts of quantum computing with multiagent learning. QMARL is presented here as an exploratory research direction, rather than as an implemented component of this thesis, where it will be further explored in the second section.

The second section presents a systematic review, conducted following the PRISMA methodology, to address the proposed research questions and provide insight into the novel aspects of the current study.

2.1 What is Machine Learning and Deep Learning?

ML is a subfield of AI as shown in Figure 2.1 dedicated to the development of algorithms capable of learning patterns from data and improving their performance without the need for explicit programming. Its origins date back to the 1950s, with the first chess programs by Turing and Champernowne, as well as the *draughts* algorithm developed by Arthur Samuel in 1955, considered the first machine learning system to gain public recognition (Shinde and Shah 2018).

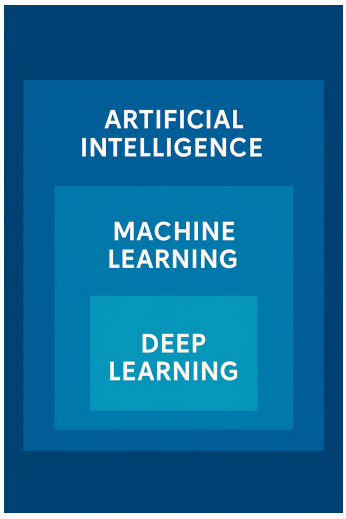

Figure 2.1: The Hierarchy of Artificial Intelligence

The evolution of ML includes fundamental milestones, such as Rosenblatt's perceptron (1957), the backpropagation algorithm popularized in the 1980s, and the introduction of Support Vector Machines by Vapnik and Cortes in 1995. These were followed by ensemble methods, among which AdaBoost (1997) and *Random Forests* (2001) stand out, increasing both the accuracy and robustness of models (Shinde and Shah 2018).

Currently, the discipline encompasses a wide range of algorithms, including logistic regression, *Naïve Bayes*, decision trees, KNN, and artificial neural networks. The availability of open-source frameworks, such as TensorFlow, Apache Spark MLib, and Shogun, has significantly contributed to the dissemination and large-scale practical implementation of these techniques (Shinde and Shah 2018).

The applications of ML span multiple domains: computer vision (object recognition and detection), prediction (medical diagnosis, network intrusion detection), natural language processing, and information retrieval. These examples highlight the central role of machine learning in advancing contemporary artificial intelligence (Shinde and Shah 2018).

Although traditional machine learning techniques remain valuable, they are limited in their ability to capture complex data representations. DL has therefore emerged as a promising subfield that addresses these limitations by employing multi-layer neural architectures capable of learning hierarchical and non-linear feature representations.

DL is a specialized subfield of ML that focuses on the use of artificial neural networks with multiple hidden layers to model complex patterns in large-scale data. The term "deep" refers to the presence of several layers of non-linear processing units that progressively extract higher-level representations from raw inputs. This hierarchical structure enables DL models to learn abstract features and capture intricate relationships in data more effectively than traditional ML techniques (Shinde and Shah 2018).

The origins of DL are rooted in the early studies of neural computation, from McCulloch and Pitts' neuron model (1943) to Rosenblatt's perceptron (1957). However, it was not until the development and popularization of backpropagation in the 1980s and subsequent advances in computational power, particularly through GPUs, that deep neural networks became feasible at scale. More recent breakthroughs include the introduction of CNNs for image recognition, RNNs and LSTMs for sequence modeling, as well as GANs for data generation (Shinde and Shah 2018).

Today, DL has become central to many cutting-edge applications. In computer vision, it powers object detection, image recognition, and medical imaging analysis. In natural language processing, DL techniques enable speech recognition, machine translation, and semantic analysis. Other domains, such as bioinformatics, drug discovery, financial prediction, and customer relationship management, also benefit from DL's capacity to handle high-dimensional and heterogeneous data. The ability of deep models to learn features automatically from raw data distinguishes them from traditional ML approaches and explains their widespread adoption in modern AI research and industry (Shinde and Shah 2018).

2.2 Fundamentals of Multi-Agent Reinforcement Learning

MARL is a specialized area of RL that focuses on environments where multiple autonomous agents interact and learn simultaneously. Unlike traditional single-agent RL, where an agent optimizes its policy in a stationary environment, MARL presents a more dynamic setting.

Each agent not only learns from its environment but must also adapt to the evolving behaviors of other agents, introducing unique challenges such as non-stationarity, partial observability, and the need for coordination (Busoniu, Babuska, and De Schutter 2008). The fact that every agent's decisions affect the learning process of others makes MARL significantly more complex but also more powerful in solving real-world problems.

In Figure 2.2, a comparison is presented between the classical RL paradigm and the MARL framework. In RL, a single agent interacts with the environment by selecting actions a_t based on the current state s_t, subsequently receiving a reward signal r_t to guide its learning process. However, in MARL, multiple agents simultaneously interact with the environment, each taking their own actions $(a_t^1, \ldots, a_t^n)$ while receiving individual observations $(o_t^1, \ldots, o_t^n)$ and rewards r_t. This fundamental difference leads to more complex dynamics, requiring additional mechanisms to handle coordination and competition effectively.

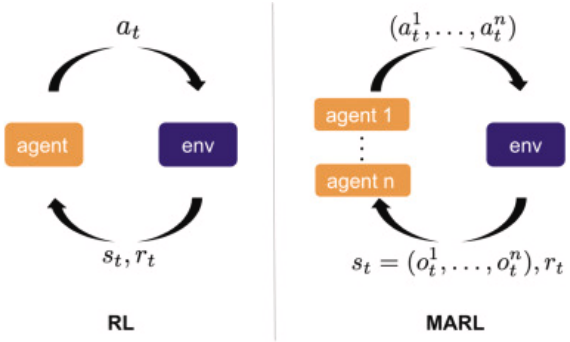

Figure 2.2: Comparison between single-agent reinforcement learning (RL) and multi-agent reinforcement learning (MARL) from Nweye et al. 2022.

One of the major challenges in MARL is non-stationarity. Since all agents are learning at the same time, the environment continuously changes from their perspective, making it difficult to converge to optimal strategies (Hernandez-Leal, Kartal, and Taylor 2019). Another challenge arises from partial observability, where each agent has limited access to global information and must make decisions based on local observations (Oliehoek and Amato 2016). Additionally, as the number of agents increases, the state-action space grows exponentially, creating scalability issues (K. Zhang, Z. Yang, and Basar 2021). Coordination and communication between agents also play a fundamental role, especially in cooperative settings where they need to share information and synchronize actions (Lowe et al. 2017).

To address these challenges, different learning approaches have been developed. IQL allows each agent to learn its own policy as if the others were part of MARL the environment, though this often leads to instability (Tan 1993). A more structured approach is CTDE, where agents receive global information during training but act independently during execution, enhancing coordination while maintaining scalability (Foerster et al. 2018). MAAC methods further improve stability by combining policy-based and value-based techniques (Lowe et al. 2017). Biologically inspired approaches, such as ACO and swarm intelligence, have also been integrated into MARL, taking inspiration from nature to create robust decentralized decision-making strategies (Dorigo and Stützle 2019).

In space exploration, MARL has emerged as a promising tool for autonomous planetary missions, where multiple robotic agents must explore unknown terrain, collect scientific data, and optimize their resources (Baker et al. 2022). By leveraging decentralized learning, robotic teams can dynamically adapt to environmental uncertainties while maximizing

efficiency. In some cases, nature-inspired strategies, such as pheromone-based path reinforcement, have been used to enhance exploration, ensuring that robots do not cover the same areas unnecessarily. This mirrors the behavior of ants, which communicate using pheromones to optimize their foraging paths (Dorigo and Stützle 2019).

By tackling the challenges of multi-agent interactions, MARL has proven to be a powerful framework for solving complex, real-world problems, especially in scenarios requiring collaboration, adaptation, and efficiency. As research continues to evolve, MARL will likely play an even greater role in advancing space robotics and autonomous exploration.

2.3 Quantum Multi-Agent Reinforcement Learning: Principles and Future Directions

This section reviews the emerging field of QMARL from a theoretical perspective. QMARL combines principles from MARL and quantum computing and is included here strictly as a literature survey to contextualize promising future directions; no quantum algorithms or QMARL implementations were conducted as part of this thesis. The primary motivation behind QMARL in the literature is to explore whether quantum techniques can improve learning efficiency and scalability (Yu and Zhao 2023;Cho et al. 2024).

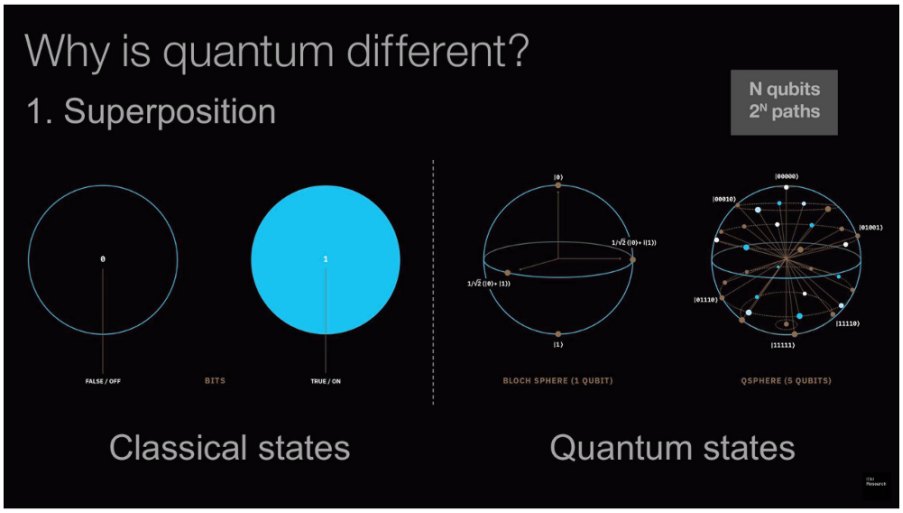

Figure 2.3: Comparison of classical and quantum states from Payments Cards and Mobile.

In Figure 2.3, the concept of superposition in quantum computing is illustrated by comparing classical and quantum states. On the left, classical bits are depicted as having distinct states of either 0 (false/off) or 1 (true/on). In contrast, the right side of the figure demonstrates how quantum states operate differently, represented by the Bloch sphere for a single qubit and the Q-sphere for multiple qubits. The Bloch sphere visualizes how a qubit can exist in a superposition of both 0 and 1, rather than being limited to a single value. The Q-sphere further extends this concept to multiple qubits, showing how they can simultaneously exist in exponentially many states. This fundamental property of quantum computing, enabled by superposition, allows quantum systems to process information more efficiently than classical systems.

In QMARL literature, quantum agents are often described as leveraging qubits and quantum properties such as superposition and entanglement to represent and process information differently than classical agents (Cho et al. 2024;Yu and Zhao 2023). Authors propose quantum-enhanced variants of policy- and value-based methods and investigate QNN's for model representation. These ideas are presented here as surveyed research and potential extensions to classical MARL; they were not implemented in the experiments or simulations of this thesis.

Another aspect frequently discussed in the literature is the conceptual use of quantum entanglement to model tight correlations between agent actions (Cho et al. 2024;Yu and Zhao 2023). While theoretically intriguing, practical use of entanglement for coordinating multi-agent policies remains speculative and is highlighted here as a potential direction for future research rather than an implemented technique in this work.

Theoretical work highlights potential advantages of QMARL relative to classical MARL: quantum representations may compactly encode large state-action spaces and quantum-assisted optimisation could, in principle, accelerate certain search procedures (Cho et al. 2024;Yu and Zhao 2023). However, these results remain largely theoretical or confined to constrained simulated studies, and practical large-scale QMARL systems are not yet available; no such implementations were developed in this thesis.

In summary, QMARL is a promising area of research that proposes to leverage quantum computing to enhance learning and decision-making in multi-agent systems. This section provides a critical overview for readers interested in future extensions; implementing QMARL is outside the scope of the current thesis and remains for follow-up work.

2.4 Technological Overview

Space exploration has advanced significantly, from early satellites to complex manned and robotic missions. These programmes have driven technological progress and expanded our understanding of the universe. Yet, challenges remain, including vast distances, extreme environments, technological limits, and communication delays with Earth (Redondo et al. 2016).

2.4.1 Technological Challenges in Space Exploration

Automation is key to addressing these issues. Remote operations and automated tasks boost mission efficiency and safety, while autonomous systems help overcome communication delays and ensure continuous operations in harsh conditions. This autonomy is critical for exploring other planets effectively (Redondo et al. 2016).

Mars rovers like Curiosity and MER have made great strides, but they face challenges on uneven terrain. The Curiosity rover, part of the MSL mission, has been a success but struggles with obstacles and requires intricate trajectory planning. With a maximum speed of 4 cm/s on flat terrain, wheeled rovers are limited by the size of obstacles they can handle and the terrain they can traverse, such as inclines over 20 degrees or rocky landscapes (Redondo et al. 2016).

Technological advancements have led to increasingly complex challenges in the field of planetary exploration. Over recent years, there has been a growing emphasis on robotic autonomy to enhance the investigation of extraterrestrial surfaces, such as Mars. Rover platforms and

autonomous drones are now designed to execute tasks like geological analysis, navigation, and mapping without the need for real-time input from Earth-based operators. However, despite the progress exemplified by rovers like Perseverance, key challenges persist in areas such as autonomous navigation and the ability to effectively navigate unknown and dynamic environments, which remain a central focus of ongoing research efforts (Zhou et al. 2024).

2.4.2 Technological Exploration

MARL involves multiple autonomous agents learning to interact and coordinate within a shared environment. This approach is particularly valuable in dynamic, uncertain, and complex settings, such as those encountered during planetary exploration missions. MARL enables these autonomous agents to efficiently collaborate and make decentralized decisions to achieve common objectives (Taghavi 2024).

The integration of MARL into robotic systems for planetary exploration, including autonomous rover fleets, has been demonstrated to enhance exploration efficiency and facilitate more effective responses to unexpected challenges (Zhou et al. 2024).

Quantum computing harnesses the unique properties of quantum-mechanical systems, such as superposition and entanglement, to perform computations significantly faster than classical computers. The integration of these quantum principles with MARL has given rise to QMARL, a novel approach that holds the potential to overcome the computational limitations of traditional methods. By leveraging quantum algorithms, QMARL can process large volumes of data in a shorter timeframe, making it well-suited for real-time decision-making in high-risk and dynamic environments, such as those encountered during space exploration missions on the Martian surface (Yu and Zhao 2023).

In the exploration of unfamiliar terrains, autonomous agents must possess the ability to adapt to novel and unanticipated situations. Traditional approaches, such as heuristic-based methods or single-agent RL, often encounter challenges in dynamic environments where obstacles and conditions are continually evolving (X. Li, Ren, and Y. Li 2024). In contrast, MARL offers a robust solution for navigating such environments. For example, teams of rovers can learn to collaboratively explore Martian landscapes, with each rover contributing to the collective understanding of the terrain, avoiding obstacles, and mapping new areas without centralized control.

Some recent studies report QMARL experiments in highly controlled simulated settings (L. Hu, Wei, and Yin 2025); these works are preliminary and typically confined to small-scale simulations or proof-of-concept hardware demonstrations. They are cited here to illustrate active research directions, but the present thesis does not reproduce or implement these quantum-enhanced approaches.

2.5 Systematic Review

This systematic review aims to examine the current state of research on the application of quantum computing and MARL for autonomous exploration of unknown environments, particularly in the context of space exploration.

2.5.1 Research Questions

As part of this systematic review, a set of central research questions has been defined to guide the analysis and to structure the critical discussion of the extant literature. The principal research questions addressed are summarised in Table 2.1.

Table 2.1: Research Questions Addressed in the Review

	Research Question
RQ1	What are the main practical applications of MARL and QMARL in solving exploration and navigation problems in dynamic and unknown environments?
RQ2	How do MARL and QMARL approaches compare to traditional methods (heuristics, single-agent RL, or classical techniques) in terms of efficiency and effectiveness?
RQ3	What recent advancements facilitate the implementation of QMARL in real-world or simulated scenarios?
RQ4	What metrics and benchmarks are used to evaluate the performance of MARL and QMARL in exploration and navigation problems?

RQ1 explores the current uses of MARL and QMARL in solving complex exploration and navigation problems in dynamic, uncertain environments like the Martian surface.

RQ2 compares the advantages and disadvantages of MARL and QMARL approaches against traditional single-agent and multi-agent methods.

RQ3 looks at the technological advancements that have enabled the practical application of QMARL in simulations and real-world.

And final question is **RQ4**, which focuses on the metrics used to evaluate the performance of MARL and QMARL in exploration and navigation tasks.

2.5.2 Data Sources

To answer these research questions, we conducted a comprehensive search of the following academic databases and search:

Table 2.2: Electronic Databases Used in the Review

ID		**Database**	**URL**
ED1		IEEE Xplore	`https://ieeexplore.ieee.org/Xplore/home.jsp`
ED2		Web of Science	`https://www.webofscience.com`
ED3		ADS	`https://ui.adsabs.harvard.edu`

The IEEE Xplore digital library was selected for its extensive coverage of technical literature in areas related to robotics, control systems, and quantum computing.

The Web of Science database provided a broad interdisciplinary source covering research across multiple fields, from computer science to physics.

The astrophysics data system(ADS) was chosen as a key source for identifying relevant research on space exploration, which could provide valuable insights into the application of advanced computing techniques, such as QMARL, to solve problems in this domain.

2.5.3 Search Terms

The search terms focused on exploration or navigation in unknown or unexplored environments, as there is a wealth of articles in this area. It also included a query that focused on the same situation but with a greater emphasis on quantum-based solutions.

The **main query** is:

("Quantum") AND (Reinforcement learning OR Multi-agent) AND (Exploration OR Navigation) AND (Unknown OR Unexplored) OR (Reinforcement Learning OR Q-learning) AND (Exploration OR Navigation) AND ("Unknown Environments" OR "Unexplored Spaces") AND (Multi-agent OR Agent)

The search seeks academic articles or documents that combine topics related to quantum computing, RL, and the exploration of unknown environments. It includes papers mentioning 'Quantum, indicating an interest in the connection between quantum computing and the other subjects. It also encompasses studies on 'Reinforcement Learning' or

'Multi-agent, covering both multi-agent and single-agent systems. The search focuses on exploration or navigation, especially in 'Unknown' or 'Unexplored' environments. To broaden the scope, the query considers references to 'Q-Learning', a RL technique, and

'Unknown Environments' or 'Unexplored Spaces. The query logic is designed to capture relevant literature exploring the intersubsection of quantum computing, intelligent navigation systems, and the exploration of unmapped spaces.

2.5.4 Inclusion and Exclusion Criteria

To ensure the relevance and quality of the selected studies, the following inclusion and exclusion criteria were applied:

Table 2.3: Inclusion Criteria (IC) Applied in the Study Selection

	Inclusion Criterion (IC)
IC1	The study must include RL or multi-agent systems (for MARL) or QRL /QMARL.
IC2	The study must address exploration, navigation, or route optimization.
IC3	It must be related to unknown, dynamic, or partially observable environments.
IC4	The study must present practical applications or relevant simulations, such as robotics, autonomous systems, logistics, or environmental exploration.
IC5	The study must be published in the last 5 years (2019–2024).
IC6	The study must be written in English.

The table 2.3 reflects our focus on studies that investigate the potential of MARL, QMARL, and related techniques in solving complex exploration and navigation problems in dynamic, uncertain environments.

The table 2.4 helps to filter out studies that are not directly relevant to the research questions or do not meet the minimum standards of quality and currency.

Table 2.4: Exclusion Criteria (EC) Applied in the Study Selection

ID	Exclusion Criterion (EC)
EC1	The study does not involve RL or multi-agent systems (MARL or QMARL).
EC2	The study does not address exploration, navigation, or route optimization.
EC3	The study is purely theoretical, without practical application or simulation.
EC4	The study is not written in English.
EC5	The study was published before 2019.
EC6	Duplicated or irrelevant sources (e.g., opinion articles, abstracts without full text).

2.5.5 Quality Assessment

The quality of the selected studies was assessed based on the following:

- Relevance to exploring unknown environments using robots or related systems, even if they don't directly address Mars or QMARL.
- Clarity of results, including clear metrics, consistent findings, and well-supported conclusions.
- Scientific impact, such as number of citations or alignment with the project's objectives.

2.5.6 Data Extraction and Synthesis

A total of 385 sources were identified through the research queries across multiple databases. After removing duplicate entries, 341 papers proceeded to the abstract screening process, where they were assessed against the inclusion and exclusion criteria. This resulted in the exclusion of 242 papers, leaving 99 papers to undergo full-text assessment.

Following the application of the inclusion and exclusion criteria at the full-text level, 21 papers were excluded, resulting in 78 eligible studies. Additional articles were identified through reference tracking, though the exact number remains to be determined.

An additional 39 articles were identified through reference tracking, where the citations of selected studies were reviewed to find relevant sources not captured in the initial database search. This step enhanced the comprehensiveness of the review. The flow diagram on figure 2.4 summarizes the overall selection process.

2.5.7 Research Questions' Answers

This subsection focuses on examining each research question and, based on the analysis performed previously, uncovering the current state of knowledge with respect to each question.

RQ1 - What are the main practical applications of MARL and QMARL in solving exploration and navigation problems in dynamic and unknown environments?

Recent literature highlights a growing interest in using MARL to address complex exploration and navigation problems. Approaches vary, but typically emphasize the ability of agents to cooperate and adapt to uncertain environments.

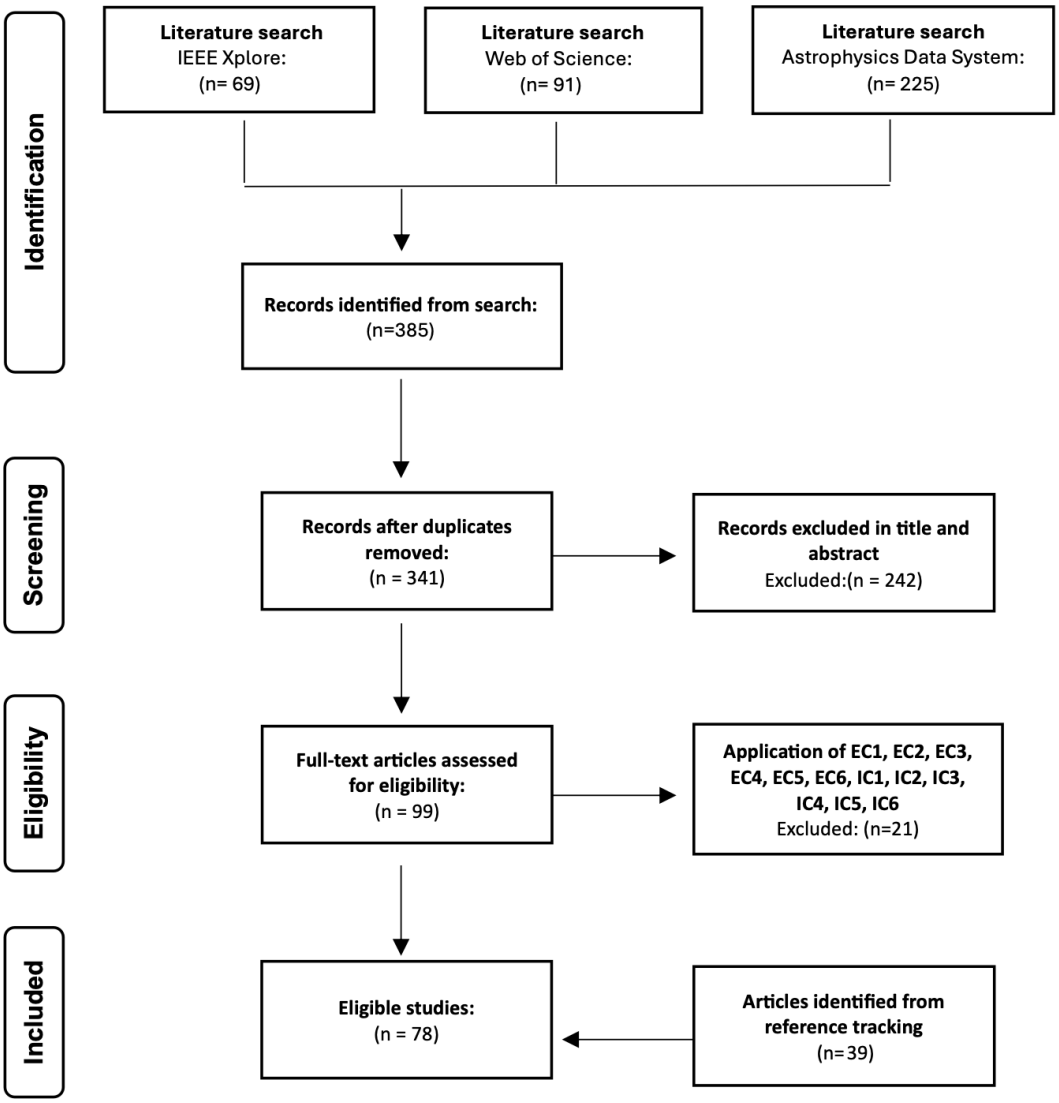

Figure 2.4: Flow diagram of the paper selection process for the systematic review

Multi-robot exploration has been investigated in several studies that have explored the use of MARL to coordinate multiple robots in the exploration of unknown environments (Liu and Wu 2019 ; J. Wang et al. 2023).

The use of multiple robots can increase efficiency and speed in the exploration of complex environments. Approaches such as MAMECS leverage attention mechanisms to reduce the state space and enhance cooperation strategies among the robots (Q. Chen et al. 2024 ; Deng, Gong, and L. Li 2022). H2GNN leverages a topological graph to model the environment, aggregating information hierarchically to optimise exploration (H. Zhang et al. 2022b).

Autonomous navigation has seen the application of MARL algorithms to train robots for navigation in dynamic environments without reliance on prior maps. These approaches enable the robots to adapt to obstacles and other dynamic elements in the surrounding (Hamann and Wölk 2022 ; Raj and Kos 2024). The use of deep neural networks allows

robots to learn navigation policies directly from raw sensory data, eliminating the need for explicit mapping and path planning(Ali et al. 2023). The approach proposed by J. Li et al. 2021 utilizes a behaviour-based architecture for navigation, which integrates objectives such as reaching designated goals and avoiding collisions. This integration helps to mitigate the complexity of the learning process.

In scenarios with limited communication between robots, decentralized architectures leveraging MARL enable agents to independently determine when to share local maps or continue exploring on their own (Calzolari et al. 2024). This decentralized approach reduces redundant data collection and optimizes the overall mapping time (Calzolari et al. 2024). Systems like the one described in Calzolari et al. 2024 use hierarchical planning, with global planning based on genetic algorithms and local navigation through RL.

In the domain of search and rescue operations, studies have highlighted the potential of MARL for addressing the complexities and dynamism inherent in such environments (Cao et al. 2024). The HMA-SAR framework incorporates a specific state and reward design, heterogeneous curriculum training, and a hybrid decision engine, improving performance in dynamic scenarios(Cao et al. 2024)

In the context of constrained exploration, MARL algorithms can be leveraged to incorporate specific requirements and constraints, such as maintaining connectivity between robotic agents during navigation in unknown environments. The study in M. Li et al. 2022 combines expert demonstrations with RL to ensure that robots maintain connectivity while exploring.

The domain of QMARL is still in its nascent stages, with relatively fewer practical implementations compared to conventional MARL approaches. However, existing research suggests that QMARL holds considerable promise for addressing exploration and navigation challenges, leveraging the unique capabilities of quantum computing principles.

Quantum algorithms have the potential to accelerate the learning process and enhance the efficiency of exploration in improving exploration and learning tasks (Kölle et al. 2024 ; Taghavi 2024). Quantum algorithms, such as the Quantum Approximate Optimization Algorithm, have been utilized to explore large solution spaces and identify approximate solutions to complex problems (Taghavi 2024). In Y. Hu et al. 2021 explores how quantum theory can be used to optimise control through RL.

Quantum communication techniques, such as the utilization of quantum channels and agent entanglement, enable agents to learn without the need for sharing local observations, thereby reducing communication overhead (DeRieux and Saad 2024). The eQMARL approach trains decentralized policies via a split quantum critic, optimizing performance without the need for centralized classical communication. This approach leverages quantum theory principles to enable agents to learn and coordinate without relying on extensive information sharing, thereby reducing communication overhead and improving the overall efficiency of the exploration and navigation tasks.(DeRieux and Saad 2024).

RL techniques can be leveraged to optimize the architecture of quantum circuits in applications that require agents to make decisions based on quantum information processing (Dai et al. 2024). The RL-QMLAS method uses a RL agent to discover optimal sequences of quantum gates, achieving high classification accuracy (Dai et al. 2024).

Then, the conclusion is that MARL and QMARL offer innovative solutions for exploration and navigation in dynamic, unknown environments. Practical MARL applications include

large-scale multi-robot exploration, autonomous navigation without prior maps, collaborative mapping with limited communication, and search/rescue operations in uncertain scenarios. While QMARL remains in early stages, it shows promise for accelerating learning, enhancing exploration efficiency, and enabling new communication/optimization techniques through quantum computing. Future challenges involve developing more robust algorithms, generalizing to diverse scenarios, and ensuring policy security and interpretability.

RQ2 - How do MARL and QMARL approaches compare to traditional methods (heuristics, single-agent RL, or classical techniques) in terms of efficiency and effectiveness?

Recent literature suggests that MARL and QMARL can outperform traditional exploration and navigation methods in various metrics, including speed, coverage, and adaptability to dynamic environments.

In **Computing and Resource Time** we find that :

1. Traditional grid-based platforms can suffer from computational bottlenecks due to the extensive training time in MARL(Zhu et al. 2024). In contrast, the MAexp platform, which uses point clouds, achieves a sampling speed around 40 times faster than existing platforms, speeding up the training process(Zhu et al. 2024).
2. The eQMARL approach exhibits a substantial decrease in the number of centralized parameters compared to traditional baseline methods, leading to a diminished computational burden. Specifically, eQMARL requires 25 times fewer centralised parameters than a classic baseline (DeRieux and Saad 2024).
3. The Dist-QTRL framework leverages parameterised quantum circuits to efficiently generate neural network parameters, achieving a poly(log(n)) reduction in the dimensionality of the trainable parameters, which results in greater computational efficiency (K.-C. Chen et al. 2024).
4. Compared to classical algorithms, methods such as Q-learning can suffer from slow learning, especially in continuous state and action spaces (Y. Hu et al. 2021). Methods such as CMD-QL and CFPA-QL try to speed up this learning by initialising Q-tables and dynamically adjusting the exploration factor (Gao et al. 2022).

In **Exploration and Convergence** we find that :

1. MARL approaches such as SOMARL, which integrate an HTN planner, exploit the action space more efficiently in sparse reward environments with traps than methods based on neural networks (Mu et al. 2023).
2. QMIX's semantic exploration (SEG) allows for more effective exploration in spaces of actions with semantic meaning, which improves performance in complex tasks (Tse and Leung 2022).
3. eQMARL, through the use of quantum entanglement, converges to co-operative strategies faster than classical and quantum baselines, achieving speed improvements of up to 17.8% in certain scenarios (DeRieux and Saad 2024)
4. In Energy Costs we think that methods such as Quantum-powered neuromorphic computing lead to a significant reduction in energy consumption(Taghavi 2024)

When examining the **efficacy** of these approaches, research indicates that:

1. Traditional methods can be limited in their ability to efficiently coordinate multiple agents in exploration (J. Hu et al. 2020a)
2. QMIX(SEG) demonstrates superior performance to other state-of-the-art MARL methods in the SMAC, indicating the method's capacity for complex, collaborative tasks(Tse and Leung 2022).
3. In addition to converging faster, eQMARL achieves a higher overall score compared to classical and quantum methods in MDP and POMDP environments (DeRieux and Saad 2024).
4. Dist-QTRL achieves a competitive performance, with average rewards similar to those of classic models, but with significantly fewer parameters, which demonstrates the framework's effectiveness in terms of model accuracy and efficiency (K.-C. Chen et al. 2024).
5. The combination of meta-heuristic algorithms (such as Simulated Annealing and Particle Swarm Optimisation) with QRL produces near-optimal results in tasks such as MiniGrid and Cart Pole(Kölle et al. 2024)

The literature indicates that MARL and QMARL approaches demonstrate significant advantages over traditional methods in terms of efficiency and effectiveness, particularly in complex tasks and multi-agent environments. MARL enhances the capacity for collaboration and exploration, while QMARL accelerates the learning process and reduces computational resource requirements. Conversely, traditional methods such as Q-learning and other heuristics can be hindered by slow learning, scalability challenges, and limited ability to manage interactions among multiple agents. Hybrid approaches that integrate classical techniques with MARL/QMARL show promise for specific scenarios, combining interpretability and optimized performance. Ultimately, the selection of the most appropriate approach depends on the specific context and application requirements; however, it is evident that MARL and QMARL provide more effective solutions for many complex scenarios compared to traditional methods.

RQ3 - What recent advancements facilitate the implementation of QMARL in real-world or simulated scenarios?

The development of quantum hardware is critical for the successful implementation of (QMARL), a cutting-edge field that combines the power of quantum computing with the complexity of multi-agent learning systems. To make this vision a reality, progress in several key areas is essential: improving the stability and coherence of qubits, increasing the number of qubits and their connectivity, and leveraging advanced quantum simulators.

Stability and coherence are among the most pressing challenges in quantum computing. Qubits, the fundamental units of quantum information, are incredibly sensitive to their environment. This sensitivity often leads to decoherence, where the information stored in a qubit is lost due to interactions with external factors. For QMARL, where algorithms require deep, complex quantum circuits, maintaining coherence over longer periods is essential (Dai et al. 2024). Recent advancements in technologies like superconducting qubits, trapped ions, and silicon-based qubits have shown promise in reducing error rates and enhancing stability. These improvements are paving the way for algorithms capable of handling the intricate demands of multi-agent learning scenarios.

Equally important is the need to scale the number of qubits and improve their connectivity. Each additional qubit expands the system's ability to represent and process information exponentially, enabling the modeling of increasingly complex environments. However, raw numbers alone aren't enough—connectivity between qubits plays a critical role in determining how efficiently quantum operations can be performed (Lockwood and Si 2020 ; K.-C. Chen et al. 2024). High connectivity allows for faster and more precise interactions between qubits, which is particularly valuable for QMARL. Operations like quantum entanglement, which enable agents to share information and coordinate strategies, become significantly more effective with improved connectivity. Advances in dynamic coupling architectures and innovative qubit network designs are driving progress in this area, enabling the execution of more powerful algorithms.

Finally, quantum simulators serve as an invaluable tool in the development of QMARL. Since current quantum computers are limited in their capacity and stability, simulators allow researchers to experiment with and fine-tune algorithms in controlled environments before deploying them on actual hardware. (DeRieux and Saad 2024). Platforms like IBM's Qiskit and Google's Cirq provide environments where algorithms can be tested, optimized, and evaluated. This not only speeds up the research process but also ensures that models are well-prepared for the constraints of real-world quantum devices.

The development of software tools, such as quantum frameworks and libraries (Qiskit, Cirq, TensorFlow Quantum) (DeRieux and Saad 2024 ; Hamann and Wölk 2022), along with simulators like OpenAI Gym (DeRieux and Saad 2024 ; Moflic, Garg, and Paler 2023) with extensions for QMARL, has played a crucial role in making QMARL more accessible. These tools allow researchers and developers to experiment with and validate algorithms in simulated environments, making it easier to implement and refine complex approaches before moving to actual quantum hardware (Y. Hu et al. 2021).

The literature highlights several key advancements that have enabled the practical implementation of QMARL. These include progress in enhancing the stability and coherence of qubits, which is crucial for maintaining the integrity of quantum information necessary for complex QMARL algorithms. Additionally, increases in the number of available qubits and improvements in their connectivity have expanded the computational capacity and facilitated more efficient coordination among multiple agents. The development of advanced quantum simulators has provided essential research and experimentation platforms for testing and optimizing QMARL algorithms prior to deployment on physical quantum hardware. Furthermore, the creation of specialized quantum algorithms tailored for MARL, leveraging unique quantum phenomena, has accelerated the learning process. The integration of classical and quantum techniques, such as the use of variational quantum circuits, has yielded powerful and flexible hybrid approaches for QMARL. The emergence of supporting software frameworks and libraries has also facilitated the implementation and deployment of QMARL systems. Finally, the demonstration of QMARL applications, including examples like variational quantum circuits and quantum neural networks, has showcased the potential of this approach in real-world scenarios. These advancements have enabled the realization of QMARL in both simulated environments and actual quantum hardware, opening up new possibilities for the development of more efficient and effective multi-agent systems.

RQ4 - What metrics and benchmarks are used to evaluate the performance of MARL and QMARL in exploration and navigation problems?

Several metrics are commonly used to evaluate the performance of MARL and QMARL in exploration and navigation tasks. One of the key indicators is area coverage, which measures the proportion of the environment that has been explored by the agents. A higher coverage suggests a more complete and effective exploration process (Q. Chen et al. 2024; Sygkounas et al. 2022). This can be assessed by calculating the percentage of visited grid cells in a discrete map (Kalra et al. 2024) or by quantifying the mapped area in a continuous space (Q. Chen et al. 2024). Another critical metric is exploration time, referring to the duration required for agents to explore a given percentage or the entirety of an environment. More efficient approaches achieve lower exploration times (Q. Chen et al. 2024).

Entropy is also frequently employed in mapping contexts to measure uncertainty or lack of knowledge about the environment. A rising entropy value over time suggests that agents are actively exploring unknown areas, while a decreasing trend indicates a gradual mapping of the space (Botteghi et al. 2021). In scenarios where hidden targets are present, the target discovery rate is a valuable measure, as it reflects how effectively agents locate these targets. A higher discovery rate indicates more efficient exploration strategies (Cao et al. 2024).

When evaluating navigation performance, several other factors come into play. Path length, for instance, measures the total distance an agent travels to reach a goal. Shorter path lengths suggest more efficient routing strategies (Y. Yang, Bevan, and B. Li 2020; H. Xie et al. 2023). Similarly, time to goal assesses the duration needed for an agent to reach its objective, with shorter times indicating more effective navigation (Y. Yang, Bevan, and B. Li 2020). Another crucial metric is the success rate, which represents the percentage of successful attempts in reaching a goal. A higher success rate implies a more reliable navigation strategy (Cao et al. 2024). Additionally, energy or resource consumption is a key consideration, particularly in scenarios where efficiency is paramount. This metric quantifies the energy or computational resources required for task completion (Taghavi 2024). Lastly, the frequency of collisions is an essential factor, as a lower number of collisions reflects safer and more effective navigation (Liu and Wu 2019; Jin et al. 2019).

To ensure rigorous evaluation and comparison of different algorithms, standardized environments and datasets serve as benchmarks. Simulated environments play a crucial role in this process. Gridworld, for example, is a simplified space where the environment is divided into discrete cells, making it ideal for testing fundamental principles of MARL and QMARL algorithms (H. Zhang et al. 2022a; Kölle et al. 2024). Tasks in Gridworld can range from full environment exploration to locating specific objectives (Karami, Aghababa, and Keyhanipour 2020). Another widely used platform is simulated robotics, where virtual robots engage in exploration and navigation tasks. These benchmarks often rely on software like Gazebo (Han, Fang, and He 2022; J. Hu et al. 2020b) and V-REP (Q. Chen et al. 2024) to create realistic simulations that account for robotic dynamics and environmental interactions. Robots in these simulations are typically equipped with sensors such as LIDAR (Alcalde et al. 2022; Ali et al. 2023) or cameras to collect data about their surroundings, and some environments also feature Ackermann vehicles equipped with radars (Zhu et al. 2024).

Other benchmark frameworks include MPE, designed for MARL(Liu and Wu 2019), and SMAC, which is particularly useful for testing MARL algorithms in combat and strategic scenarios (Tse and Leung 2022). MiniGrid is another noteworthy environment, primarily used for navigation and reinforcement learning tasks, as well as for optimizing quantum circuit parameters (Kölle et al. 2024; K.-C. Chen et al. 2024).

The combination of these performance metrics and standardized benchmarks is essential for assessing the effectiveness of MARL and QMARL algorithms in exploration and navigation tasks. The use of platforms such as Gridworld, robotic simulation environments like Gazebo and V-REP, and frameworks such as MPE and SMAC, along with real-world datasets like KITTI and university campus data, enables comprehensive evaluation and comparison. These rigorous assessments play a crucial role in advancing multi-agent systems and robotics research.

2.5.8 Synthesis and Answers to the Research Questions

Based on the systematic review and the analysis of the state of the art presented in this chapter, we summarize below direct answers to the four research questions (RQ1–RQ4).

RQ1 — Practical applications of MARL and QMARL: The literature indicates that MARL already has consolidated practical applications in multi-robot exploration, autonomous navigation without prior maps, collaborative mapping under limited communication, and search-and-rescue scenarios (e.g., Liu and Wu 2019; Calzolari et al. 2024; Cao et al. 2024). QMARL emerges as a nascent area with proof-of-concept demonstrations in simulation; however, its practical applications remain preliminary and are generally confined to simulated studies or small-scale demonstrations (e.g., L. Hu, Wei, and Yin 2025).

RQ2 — Comparison with traditional methods: In complex multi-agent tasks, MARL tends to outperform classical approaches (heuristics or single-agent RL) in metrics such as coverage, adaptability, and robustness (see section on RQ2). Proposals for QMARL report theoretical advantages in efficiency and representational dimensionality under restricted contexts (e.g., DeRieux and Saad 2024; K.-C. Chen et al. 2024), yet these findings remain incipient and have not been generalized to real-world scenarios.

RQ3 — Advances enabling QMARL: The main developments that may render QMARL feasible include improvements in qubit coherence and stability, increased qubit count and connectivity, and the maturation of simulators and frameworks (Qiskit, Cirq, TensorFlow Quantum). However, these advances are necessary but not sufficient: many studies remain confined to simulated environments and lack replication on scalable hardware (section RQ3).

RQ4 — Metrics and benchmarks used: Evaluations employ a combination of metrics (area coverage, exploration time, target discovery rate, path length, time-to-goal, success rate, resource consumption, and collision frequency) and standardized benchmarks (Gridworld, MPE, SMAC, Gazebo, MiniGrid, datasets such as KITTI). These metrics enable rigorous comparisons among classical approaches, MARL, and preliminary QMARL studies (section RQ4).

In summary, the review shows that MARL constitutes a mature solution for several multi-robot exploration problems, whereas QMARL represents a promising but still largely theoretical and experimental direction. For this thesis, implementation and evaluation focus on MARL in simulated environments; QMARL is discussed as future work and as a potential solution for specific challenges.

Chapter 3

Methods and Materials

This chapter details the methodological approaches and materials utilized in the development and execution of the experiments. It describes the dataset selected and the preprocessing techniques applied to it. Furthermore, it outlines the experiments conducted and the validation methods employed. Additionally, it addresses the ethical and security considerations taken into account during the data handling process.

3.1 Method and tools

To achieve the goals of this thesis, a systematic approach involving a combination of advanced computational tools, programming frameworks, and algorithmic strategies was employed. The methods and materials are described in detail below to ensure the reproducibility and transparency of the research process.

3.1.1 Map Construction and Simulation Framework

The initial step involved building a two-dimensional Mars simulation environment. This was accomplished using the Python programming language, chosen for its versatility. A grid-based map was created to simulate each possible step that rovers could take, ensuring a high level of granularity in the simulation. This grid was designed to be dynamic and adaptive, enabling it to be populated with realistic scenarios and constraints.

To increase the realism of the simulated environment, AI techniques were integrated. These techniques provided adaptive behaviors and decision-making capabilities within the simulation, contributing to a 2D Mars simulator that was as realistic as possible with the data provided.

3.1.2 MARL Environment

The next stage involved populating the simulation with rover agents implemented in Python. Rovers were categorized into two primary types: explorers and transporters, each performing distinct but interrelated tasks. Explorers were tasked with data collection and environmental mapping, while transporters optimized resource delivery and logistics within the simulation.

An algorithmic approach was implemented to optimize the transporters' routes, ensuring efficient navigation and task completion. The design and optimization of these routes were informed by prior studies in the literature on path optimization in multi-agent systems.

3.2 Dataset

Over the years, multiple Martian missions have deployed weather stations to study the planet's atmosphere, each contributing valuable insights into its climate dynamics. From the early Viking landers to the state-of-the-art MEDA system aboard Perseverance, these instruments have evolved significantly, improving our ability to monitor and understand Martian weather patterns.

Each weather station was designed with specific mission goals in mind. Some, like the Viking landers, provided the first-ever direct atmospheric measurements, while others, like Phoenix, focused on studying regional conditions. The Curiosity rover's REMS system expanded environmental monitoring, and InSight, though primarily a seismic mission, contributed atmospheric data as well. The most advanced system to date, MEDA, offers continuous, high-resolution monitoring, providing crucial data for future exploration and potential human missions.

The table 3.1 compares these weather stations across key aspects, including the range of measured parameters, temporal resolution, data availability, specialization for Martian climate studies, accessibility, and integration with other datasets. This comparison highlights how each system has contributed to our growing understanding of Mars, with MEDA standing out as the most comprehensive and advanced weather monitoring system yet.

After additional research, the MEDA dataset was identified as a reliable source of Mars weather data. This dataset is a recent contribution from NASA's Mars 2020 mission, which was not included in the previous literature review. This discovery reflects the rapid progress in planetary exploration, with new data and insights being continuously generated.

The MEDA dataset provides detailed, real-time measurements of Martian atmospheric conditions, including temperature, wind speed and direction, atmospheric pressure, humidity, solar radiation, and the concentration of atmospheric dust. Collected by the Perseverance rover in the Jezero Crater, these observations offer an unprecedented level of precision and context for understanding the dynamic weather patterns of Mars (Rodriguez-Manfredi et al. 2021).

The data used is organized into different levels of processing, each serving a specific purpose:

1. Raw Data (data_raw_env): These are the original digital values captured by the sensors, with no processing applied. They serve as the foundation for all subsequent stages and are essential for validating measurements directly at the hardware level.
2. Partially Processed Data (data_partially_processed_env): At this stage, the data has undergone initial processing and has been converted into basic physical quantities, such as voltages, intensities, and resistances. These data are mainly used to monitor sensor performance and assess their health.
3. Calibrated Data (data_calibrated_env): These data provide environmental magnitudes such as pressure, temperature, humidity, and wind. They are adjusted using pre-landing calibrations and post-landing observations, with minimal assumptions about external conditions.
4. Derived Data (data_derived_env): These are the most processed data products, created from calibrated data using models and additional assumptions. They offer consolidated magnitudes, ideal for more advanced scientific analyses, such as wind behavior or pressure variations.

Table 3.1: Comparative Analysis of Martian Weather Stations.

	Viking (1976)	Mars Pathfinder/Sojourner (1997)	Phoenix (2008)	Spirit/Opportunity (2004--2018)	REMS -- Curiosity (2012--present)	InSight (2018--present)	MEDA -- Perseverance (2021--present)
Parameter Range	Basic: temperature, pressure, wind	Simple meteorological data, focused on landing demonstration	Regional measurements, limited coverage	Complementary data supporting geological studies	Moderate range: temperature, pressure, humidity, radiation	Primarily seismology; meteorological data are secondary	Comprehensive: temperature, pressure, wind speed and direction, radiation, dust
Temporal Resolution	Sparse collections and low frequency	Point data, no continuous monitoring	Moderate resolution, limited to the mission duration	Intermittent records	Good daily resolution, though some intervals may limit capturing all dynamics	Moderate resolution, focused on the surface-atmosphere interface	High continuous temporal resolution, capturing diurnal and seasonal variations
Data Currency	Historical data, used as a reference	Data from the mission period; not updated	Historical and regional data	Data limited to the operational period	Updated data, but focused on the rover□s exploration area	Current data, with an emphasis on seismology	Continuous, real-time updated data
Martian Environment Specialization	First generation measurements, no dedicated instrumentation	Instrumentation designed for landing demonstration	Designed for regional conditions	Primary focus on geology, with supporting environmental data	Designed for environmental analysis, integrated with multiple mission objectives	Mainly focused on seismology	Specifically developed for comprehensive monitoring of the Martian environment
Accessibility and Documentation	Available in historical repositories, with limited documentation	Data available, but with reduced documentation	Data and documentation available, though with technological limitations of the time	Fragmented data and dispersed documentation	Easily accessible via NASA portals, though with a local focus	Documentation primarily oriented towards seismology	Highly accessible, with detailed documentation and community support
Integration with Other Measurements	Limited integration; isolated data	Isolated data, with no broad integration	Useful data, but not integrated into a larger system	Complementary measurements, but with limited integration among parameters	Moderate integration, considering the mission's multiple objectives	Integration focused mainly on correlating pressure with seismicity	Excellent integration with a wide range of sensors, providing contextualized analysis

To achieve a complete dataset with all the necessary information, data from the different datasets mentioned above will be combined and analyzed together.

Within these datasets, the data is further divided by SOL, a term used in space exploration to refer to a Martian solar day. A SOL is slightly longer than an Earth day, lasting approximately 24 hours and 39 minutes. This division by SOL is crucial for organizing and analyzing data from Mars, as it ensures that measurements align with the planet's daily cycles.

3.2.1 MEDA Sensors

The MEDA system is equipped with multiple sensors to study the Martian environment. The WS has two detectors (WS1 and WS2) on separate booms to measure wind speed and direction. The ATS includes five detectors, three on the RSM and two on the rover's front body to monitor temperature.

The TIRS measures thermal radiation from the surface and atmosphere, while the HS tracks water vapor levels. The RDS measures solar radiation and dust presence, and the PS, located inside the ICU, monitors atmospheric pressure (Rodriguez-Manfredi et al. 2021).

Radiation and Dust sensor

The Radiation and Dust sensor, as shown in Figure 3.1, is mounted on the rover top deck and comprises eight upward viewing UV photodetectors, eight lateral viewing UV photodiodes, a reference dark-current photodiode, and an upward looking camera. The signals coming from these photodiodes and thermopiles will be routed to the ICU to be conditioned and digitized inside the ICU. The ICU will also control the camera through its power and data interfaces (Rodriguez-Manfredi et al. 2021).

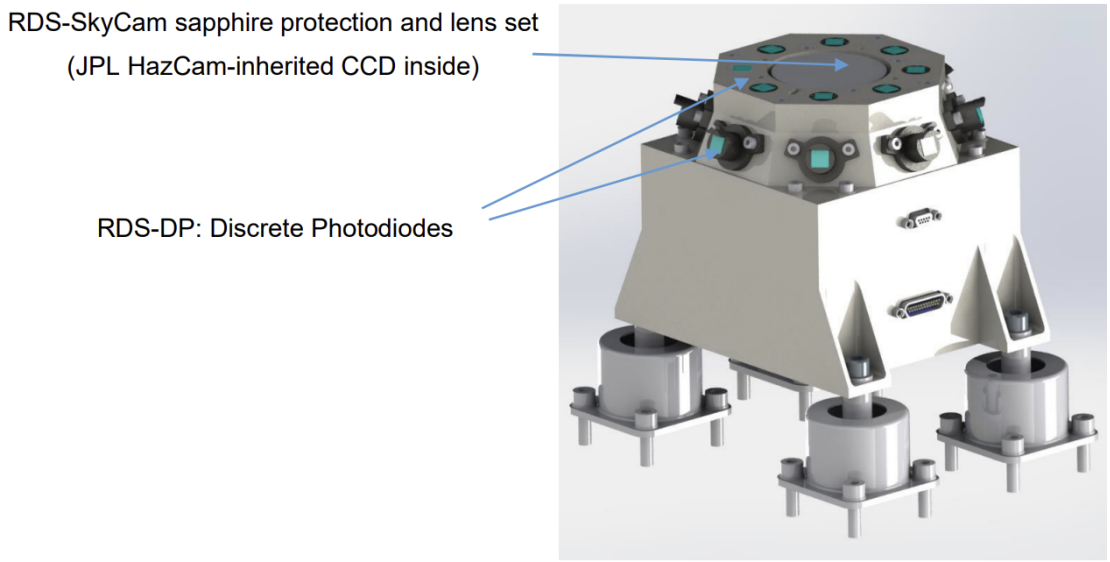

Figure 3.1: RDS illustration from Rodriguez-Manfredi et al. 2021

The RDS sensor generates the following table shown in table 3.2:

Table 3.2: Data generated by the RDS sensor from NASA Planetary Data System (PDS) 2020.

LTST	RDS_LAT_1_HG	RDS_LAT_2_HG	RDS_LAT_3_HG	...
0082 23:52:37	-0.000000000006817	-0.00008168	0.00006455	...
0082 23:52:38	-0.000000000002725	-0.00005170	0.00005174	...
0082 23:52:39	-0.000000000007199	-0.00008404	0.00005757	...

The MEDA dataset provides the LTST time for each corresponding Martian SOL and the timestamp referenced to Earth time. Additionally, the dataset includes multiple tables corresponding to the photodiodes mentioned in the sensor description. These tables will have 8 columns for the horizontal photodiodes and 8 columns for the vertical photodiodes. This values are measured in W/m2 (Rodriguez-Manfredi et al. 2021).

Thermal-Infrared Radiation Sensor

The TIRS is also mounted on the RSM and it is composed of 5 thermopiles pointing upward and downward to measure different ground and atmosphere temperatures in different infrared bands and the solar radiation reflected on the ground (albedo) as shown in figure 3.2.

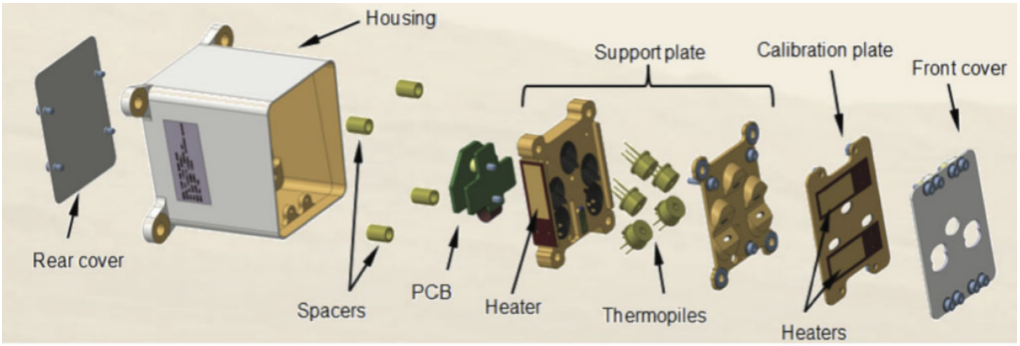

Figure 3.2: TIRS illustration from Rodriguez-Manfredi et al. 2021

The TIRS sensor generates the following table shown in table 3.3:

Table 3.3: Data generated by the TIRS sensor from NASA Planetary Data System (PDS) 2020.

LTST	AIR_TEMP	GROUND_TEMP
0082 23:52:37	207.36	202.77
0082 23:52:38	207.79	202.75
0082 23:52:39	207.20	202.69

The table 3.3 includes the SOL, which has been previously referenced, as well as two additional columns - one displaying the air temperature and the other exhibiting the ground temperature. These measurements are recorded in Kelvin (Rodriguez-Manfredi et al. 2021).

Wind Sensor

The wind sensors are housed in two small Booms structures mounted orthogonal to the RSM of the Rover. Each Boom provides 6 wind sensor transducer boards on the head of

the main boom cylinder. Booms include front-end mixed ASIC to condition and acquire the data from the wind sensors and to communicate serially with the ICU.

Figure 3.3: Wind Sensor illustration from Rodriguez-Manfredi et al. 2021

In figure 3.3 shown design concept of sensor where green plates are the wind sensor transducer boards.

The Wind Sensor generates the following table shown in table 3.4:

Table 3.4: Data generated by the Wind sensor from NASA Planetary Data System (PDS) 2020.

LTST	HORIZ_WIND_SPEED	VERTI_WIND_SPEED	WIND_DIRECTION
0082 23:52:37	6.61		287.16
0082 23:52:38	5.83		287.16
0082 23:52:39	5.88		287.16

The table 3.4 contains the SOL, which has been mentioned earlier, along with three further columns. These additional columns provide the horizontal wind speed, the vertical wind speed, and the wind direction. The wind speed measurements are in m/s, and the wind direction is in degrees (Rodriguez-Manfredi et al. 2021).

Pressure Sensor

The pressure sensor is mounted in conjunction with the Instrument Control Unit analog module, but it also utilizes a dedicated conduit to interface with the external Martian environment. This conduit traverses through the ICU base plate, the RAMP, and terminates in a cavity within the rover's top deck. The aperture is safeguarded against dust intrusion and for planetary protection purposes by a cover affixed to the rover's top deck.

The Pressure Sensor generates the following table shown in table 3.5:

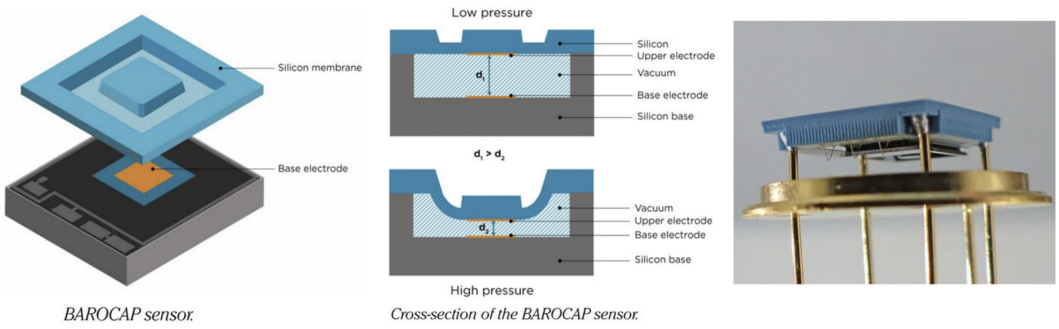

Figure 3.4: Pressure sensor illustration from Rodriguez-Manfredi et al. 2021

Table 3.5: Data generated by the Pressure sensor from NASA Planetary Data System (PDS) 2020.

LTST	PRESSURE
0082 23:52:37	758.18
0082 23:52:38	758.23
0082 23:52:39	758.20

The table 3.5 includes the SOL, as previously mentioned, and an additional column displaying the atmospheric pressure. The pressure measurements are recorded in Pascals (Rodriguez-Manfredi et al. 2021).

Humidity Sensor

The HS onboard Perseverance employs the Humicap® RSH045, which measures relative humidity (0–100% RH) through changes in the capacitance of a humidity-sensitive polymer as shown in Figure 3.5. This polymer reacts even when unpowered, enabling near-instantaneous readings after activation. Compared to earlier models, such as the RS92 Humicap® used in the REMS instrument aboard Curiosity, the RSH045 provides significant improvements: higher capacitance, up to nine times larger dynamic range, and an integrated PT1000 platinum resistance thermometer, which ensures precise measurements of the sensor chip temperature.

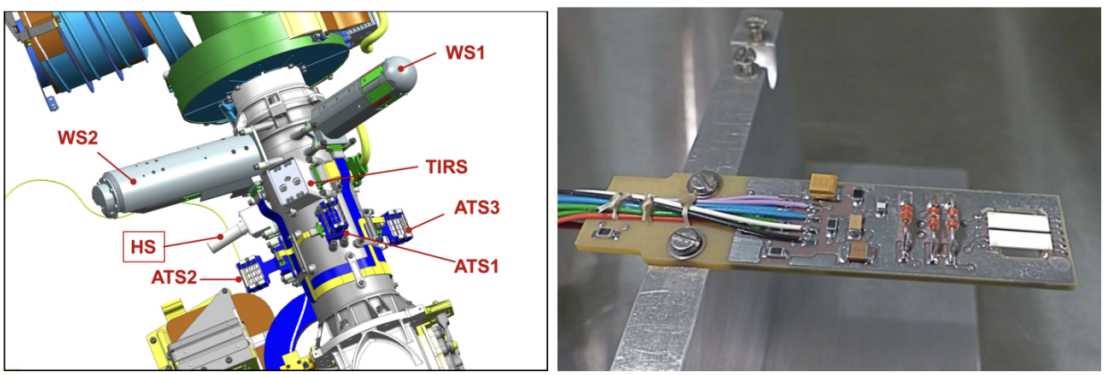

Figure 3.5: Humidity sensor illustration from Rodriguez-Manfredi et al. 2021

The sensor's performance is strongly temperature-dependent. At lower temperatures, its response time increases considerably—for example, at –70 °C the response time is approximately 450 seconds. Since the HS is constantly exposed to the Martian environment, with temperatures ranging between about –90 °C and 10 °C, measurements at very low temperatures represent an average over several minutes. Furthermore, because Humicap® capacitance is affected by CO2, calibration must be performed under low-pressure CO2 conditions to ensure accuracy (Rodriguez-Manfredi et al. 2021).

The Humidity Sensor generates the following table shown in table 3.6:

Table 3.6: Data generated by the Humidity sensor from NASA Planetary Data System (PDS) 2020.

LTST	HUMIDITY
0926 23:52:37	2,015357142857143
0926 23:52:38	1,8348809523809524
0926 23:52:39	1,8536666666666664

The table 3.6 includes SOL, as mentioned earlier, and an additional column displaying humidity. Humidity measurements are recorded as a percentage.

3.2.2 Data Processing and Integration

To ensure a comprehensive and accurate analysis of the data, it will be necessary to merge the datasets from all sensors into a unified dataset. This integration will be performed using the LTST column as the key for joining, as it serves as a common reference point across the different datasets. By aligning the data in this way, we can ensure that all sensor measurements are properly synchronized and analyzed collectively.

For the RDS sensor data, a more refined approach will be applied to derive a value that is as close as possible to the actual measurement. Specifically, we will calculate an average of the sensor readings horizontally, which we will refer to as α. This value will be assigned a weight of 0.4 to reflect its relative importance. In addition, we will calculate a vertical average of the data, referred to as β, which will carry a weight of 0.6. These weights are chosen to prioritize the vertical average slightly more than the horizontal one, based on the characteristics of the sensor data.

The final value, RDS_T, will then be calculated by combining these two weighted averages, using the formula:

$RDS_T = \alpha * 0.4 + \beta * 0.6$

This method ensures that the resulting value incorporates both dimensions of the data in a balanced way, leading to a more reliable representation of the actual sensor readings.

3.3 Data Protection, Security and Ethics

Maintaining ethical and secure data practices is paramount, particularly in research endeavors that simulate real-world systems. While this project utilizes artificially generated data modeled after the Mars Environmental Dynamics Analyzer from NASA's Perseverance rover,

it is crucial to align all data management practices with ethical and legal frameworks, such as the Artificial Intelligence Act of the European Union.

The dataset in this project does not contain personal or identifiable information. It consists entirely of simulated data designed to replicate scenarios in unknown terrain exploration.

This includes:

- Terrain mapping: Coordinates and environmental features.
- Simulated sensor readings: Data from LIDAR, proximity sensors, and environmental conditions.
- Agent activity logs: Trajectories, decisions, and exploration history.

Even though the data is fully artificial, the project adopts best practices in data protection to maintain transparency and ethical standards, ensuring compliance with the AIA's principles of responsible AI development.

3.3.1 Compliance with the Artificial Intelligence Act

The AIA sets stringent guidelines for AI systems, particularly those classified as high-risk, to ensure transparency, safety, and respect for fundamental rights. To comply with these regulations:

In Data Minimization, the dataset is carefully designed to include only the information necessary for the simulation and RL training. Any irrelevant or excessive data is excluded.

In Transparency the methodology for generating and using the dataset is clearly documented, with any limitations or uncertainties openly discussed to maintain transparency throughout the research process.

In Purpose Limitation the data is exclusively used for training and validating the multi-agent system for exploration. There is no repurposing of data, adhering to the AIA's principle of purpose limitation.

In last point, Controlled Environment all experiments are conducted in a secure, virtual simulation environment, ensuring there is no risk of impacting external systems or handling sensitive data.

3.3.2 Data Security and Ethical Considerations

Although the dataset used in this project is entirely simulated, it is handled with the same ethical care and rigor as real-world data. To ensure responsible data management, the project incorporates Privacy by Design principles, embedding data protection measures into every stage of development. All data is securely stored, with access restricted to authorized team members, and the dataset, along with the algorithms, is shared transparently to allow other researchers to replicate and validate the findings.

Beyond the technical implementation, this project recognizes the broader ethical and societal implications of applying AI to real-world scenarios like planetary exploration. By adhering to ethical and legal standards, the research ensures that its contributions are not only scientifically innovative but also trustworthy, fostering responsible AI applications in other critical and challenging domains.

This project demonstrates a strong commitment to ethical standards and compliance with the Artificial Intelligence Act. Through practices like data minimization, transparency, and secure handling, it ensures that the research is both robust and responsible. By prioritizing ethical data management, the project reinforces the credibility of its results and sets an example for future studies exploring the use of AI in extreme and innovative environments.

3.4 Experimentation

This section presents a comprehensive description of the experimental setup, detailing the simulation environment, agent strategies and the performance metrics used to evaluate the system. The overall pipeline of the project is illustrated in Figure 3.6.

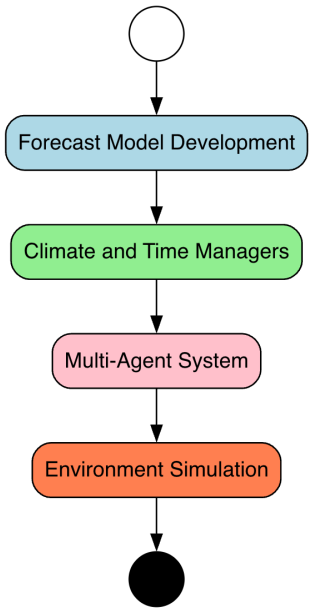

Figure 3.6: Pipeline diagram

Prior to practical implementation, a conceptual pipeline was defined to guide the organisation of the experimental process. The pipeline establishes a clear logical sequence between the system modules, ensuring methodological consistency and providing a solid foundation for the implementation phase.

The process begins with Forecast Model Development, which is responsible for preparing predictive models based on environmental data from the MEDA instrument. This module encompasses data collection, exploratory data analysis, and the development of predictive models capable of identifying climatic patterns that are relevant to the simulation.

Outputs from the forecasting module are integrated into the Climate and Time Managers, which serve as the environmental management layer. This component enables the simulation of typical Martian adverse events (e.g. dust storms and seasonal variations) and handles SOL's, thus ensuring temporal coherence within the simulated environment.

Subsequently, the Multi-Agent System module defines the coordination logic for virtual robots. It specifies agent classes (such as explorers and transporters), establishes communication protocols, and implements mechanisms for cooperation and competition. This arrangement permits the simulation of emergent behaviours and the evaluation of multi-agent coordination strategies prior to their deployment in the simulator.

Finally, the Environment Simulation module provides the integrated virtual space in which all components operate. The environment is modelled on a two-dimensional grid representing the Jezero Crater surface and supports the introduction of obstacles, samples, and heterogeneous terrain. At this stage, agents interact with the environment, execute exploration and transport tasks, and the system records performance metrics for subsequent analysis.

Together, these four modules constitute an experimental pipeline that underpins the practical implementation. This modular approach facilitates validation of inter-component dependencies, anticipates potential limitations, and enables the definition of consistent evaluation metrics for simulator testing.

3.4.1 Simulation Environment Setup

The simulation is implemented in Python and features a high-resolution grid to replicate a realistic Martian terrain. Each cell in the grid is enriched with multiple environmental attributes, including radiation, wind, temperature, and crater presence, among others. The environment dynamically adjusts these attributes based on predefined models and real-time interactions with agents. A neural network, trained on the MEDA dataset, is employed to predict the state of adjacent cells, enabling smooth transitions and realistic environmental dynamics. Additionally, the simulation incorporates adverse event managers, such as dust storms and solar radiation events, to challenge agent strategies and enhance realism.

3.4.2 Agent Strategies and Implementation

Two classes of agents are proposed to interact with the simulation: explorers and transporters. The descriptions below present high-level roles and intended behaviours; specific algorithms or sensor models will be selected during implementation and are therefore described here at a conceptual level.

- **Explorer Agents:** Explorer agents are intended to traverse the simulated terrain to identify points of interest and incrementally build an environmental map. Their behaviour focuses on systematic coverage, obstacle avoidance, and local sensing to update the shared map. Explorers will also include simple safety behaviours to retreat or seek shelter when environmental hazards are detected.
- **Transporter Agents:** Transporter agents are responsible for collecting identified samples and delivering them to a designated base or depot. Their behaviour will prioritise reliable navigation and dynamic task assignment, adapting routes and priorities as new tasks arise or conditions change. Transporters will follow safety constraints during adverse events and defer or reroute operations as needed.

Chapter 4

Implementation and Analysis

This chapter describes the implementation and evaluation of the proposed multi-agent exploration and sample-retrieval system within a possible Martian simulation. It outlines the construction of the simulated environment and the preparation of observational data used to drive it, the design of autonomous agent behaviours and coordination mechanisms, and the experimental protocol adopted to assess system performance. Emphasis is placed on how the components interact in operation — from local sensing and task generation to coordinated task allocation and emergency response — and on the metrics used to quantify effectiveness, robustness and adaptability.

4.1 Understand and pre-processing the data

The first step towards a realistic simulation is to characterise the available observational data. For this purpose, the MEDA dataset referenced in Chapter 3.2 was adopted. The dataset, provided via the Planetary Data System, comprises meteorological measurements from Mars, including temperature, pressure and wind speed among other variables. The sample shown in Table 4.1 was assembled by aggregating multiple CSV files from the instrument sensors into a single consolidated file for subsequent processing.

Table 4.1: Example of simulated environmental data

SOL	TIME	AIR TEMP	GROUND TEMP	PRESS-URE	LOCAL RELATIVE HUMIDITY	RDS_T	HORIZON-TAL WIND SPEED
1	15:25:00	12.66	-82.03	715.83	–	7.86	–
1	15:30:00	5.92	-83.31	715.62	–	7.69	–
1	15:35:00	–	–	715.33	–	7.35	–
1	15:40:00	–	–	715.21	–	7.04	–
1	15:45:00	–	–	715.02	–	6.74	–

A preliminary inspection of the dataset revealed substantial missing values. To quantify data completeness we computed feature-wise statistics summarised in Table 4.2, which reports the total count, number of missing entries, percentage missing, and data type for each variable. The horizontal wind speed variable exhibits the highest proportion of missing values (approximately 84%), while most remaining variables show missingness below 10%. Although a variable with such a high fraction of missing data might typically be discarded, wind speed was retained because of its relevance to the simulation scenarios and will be addressed during preprocessing.

Table 4.2: Descriptive statistics on the quality of environmental data

FEATURE	COUNT	MISSING	MISSING PCT (%)	DTYPE
SOL	192131	0	0.00	int64
AIR_TEMP	188918	3213	1.67	float64
GROUND_TEMP	188910	3221	1.68	float64
PRESSURE	190532	1599	0.83	float64
LOCAL_RELATIVE_HUMIDITY	179366	12765	6.64	float64
RDS_T	191018	1113	0.58	float64
HORIZONTAL_WIND_SPEED	29797	162334	84.49	float64

In the context of the simulation, the initially available variables did not provide a clear temporal reference, which is essential for analysing patterns and behaviour across different phases of the Martian environment. To address this limitation, a relative time variable denoted **SEASON** was introduced to categorise events according to Martian seasons.

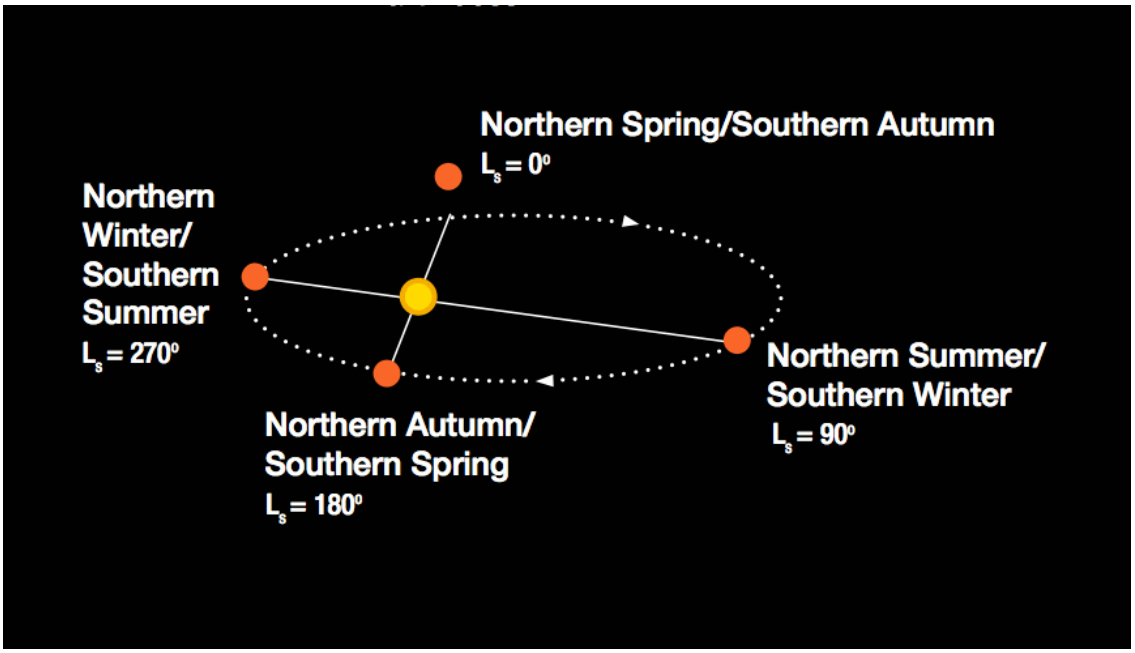

Figure 4.1: Martian seasons in terms of solar longitude (Ls). Image credit: NASA/JPL-Caltech

Season assignment was based on the LS as shown in figure 4.1, an astronomical parameter that denotes Mars' orbital position along its elliptical path around the Sun.

Then we have the `Skyfield` library, developed in Python, is a tool designed for high-precision astronomical calculations. It enables the determination of the positions of planets, stars, artificial satellites, and other celestial bodies at different points in time, making use of ephemeris data provided by the JPL of NASA.

In comparison with earlier libraries, such as `PyEphem`, `Skyfield` stands out for its greater accuracy and flexibility in handling astronomical coordinate systems, as well as for its support of Two-Line Element orbital data. Owing to its integration with libraries such as `NumPy`, it offers efficiency in computations, with applications ranging from amateur astronomy to scientific research.

Thus, `Skyfield` constitutes an accessible and reliable solution for the simulation and analysis of astronomical phenomena in both academic and professional contexts.

The computation of *LS* relies on accurate ephemerides (e.g. the `de421.bsp` file) and was implemented using the `Skyfield` library, which is widely used for precise astronomical calculations.

Math formula:

$$LS = \nu + \omega$$

Where:

- ν is the true anomaly, computed from the mean anomaly (M) and the orbital eccentricity (e) by solving Kepler's equation.
- ω is the argument of perihelion, which specifies the orientation of the elliptical orbit in space.

Season assignment is performed according to intervals of *LS*, as summarised in Table 4.3. For instance, Spring corresponds to $0° \leq LS < 90°$, Summer to $90° \leq LS < 180°$, and so on.

Table 4.3: LS values per SOL in the Spring season

SOL	SEASON	LS
1	Spring	5.41
2	Spring	5.41
3	Spring	5.42
4	Spring	5.42
5	Spring	5.42
6	Spring	5.42
7	Spring	5.42
8	Spring	5.43
9	Spring	5.43

Table 4.3 reports the *LS* values for the initial *SOLs* of Martian year 36, corresponding to the early phase of the Perseverance mission. The Martian calendar was used as a reference Turun Ursa 2021 to map seasons and the associated days, enabling the extrapolation of *LS* values to subsequent *SOLs*.

In addition, Table 4.4 lists the dates of equinoxes and solstices for Martian years 36 through 40, based on astronomical calculations Cantor, James, and Calvin 2010. These dates are essential for correlating observed climatic events and environmental patterns with the different phases of the Martian year.

Based on these tables and calculations, the **SEASON** variable was integrated into the simulation model, enabling a more detailed and context-aware analysis of Martian climatic and environmental patterns.

4.1.1 Exploratory Data Analysis of MEDA dataset

With the primary dataset consolidated, we proceeded to perform a focused exploratory data analysis. This step aims to characterise the available measurements, identify their statistical properties, and assess their suitability for simulation-driven experiments. The figures below

Table 4.4: Dates of equinoxes and solstices during Martian years 36 to 40 Cantor, James, and Calvin 2010

Martian Year	**Spring Equinox** ($LS = 0°$)	**Summer Solstice** ($LS = 90°$)	**Autumn Equinox** ($LS = 180°$)	**Winter Solstice** ($LS = 270°$)
36	02-07-2021	25-08-2021	24-02-2022	21-07-2022
37	26-12-2022	12-07-2023	12-01-2024	07-06-2024
38	12-11-2024	29-05-2025	29-11-2025	25-04-2026
39	30-09-2026	16-04-2027	17-10-2027	12-03-2028
40	17-08-2028	03-03-2029	03-09-2029	28-01-2030

illustrate the main visualisations used to examine the data distribution, detect anomalies, and inform preprocessing choices for the simulation model.

We begin with a correlation plot that highlights pairwise linear relationships among variables and helps to identify which measurements are strongly associated.

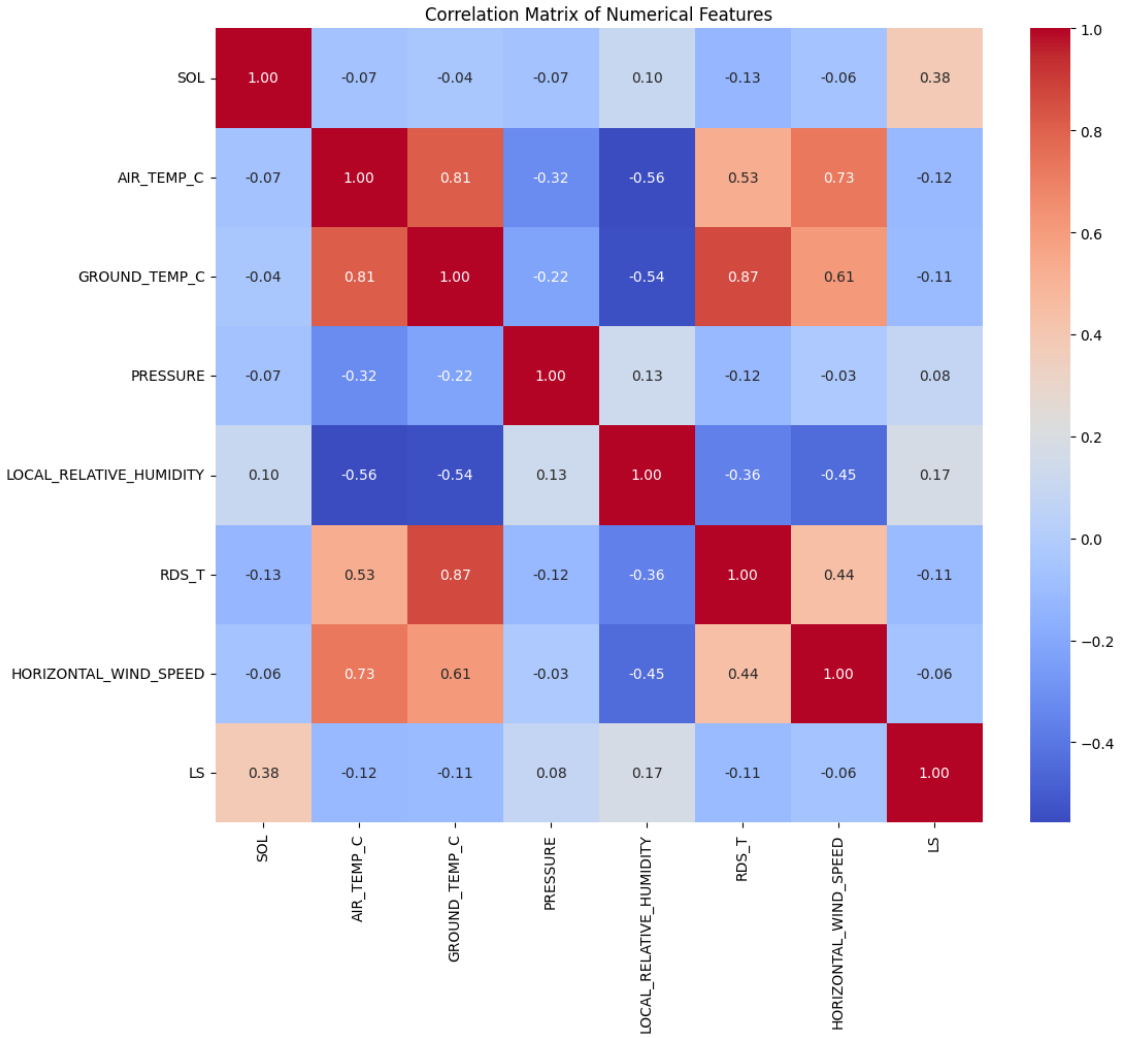

Figure 4.2: Pearson correlation matrix of the MEDA dataset

The correlation matrix provides in Figure 4.2 a quantitative assessment of linear interdependencies among the meteorological variables measured on Mars. Pearson's correlation coefficient was used as the metric, ranging from -1 to 1. Positive coefficients near 1 indicate a strong direct linear relationship, while negative coefficients near -1 indicate a strong inverse relationship. Values close to zero suggest little or no linear dependence between the variables.

A strong positive correlation is observed between `AIR_TEMP_C` (air temperature) and `GROUND_TEMP_C` (surface temperature), with a Pearson coefficient of 0.81. This result aligns with expected thermal behaviour on Mars, where incident solar radiation simultaneously warms the surface and the lower atmospheric layers. Although the Martian atmosphere is thin and primarily composed of carbon dioxide, conductive and radiative heat exchange between the ground and the air still occurs and can account for this elevated correlation.

`GROUND_TEMP_C` also shows an even higher correlation with `RDS_T` (0.87), which is plausibly associated with surface thermal radiation or heat flux from the ground. This suggests that `RDS_T` captures a thermal component that depends directly on surface temperature, as would be expected in an environment with low atmospheric thermal inertia.

There is also a notable positive correlation between `AIR_TEMP_C` and `HORIZONTAL_WIND_SPEED` (0.73). On Mars, wind patterns are strongly modulated by thermal gradients arising from uneven solar heating, particularly near the terminator (the day–night boundary). The positive association indicates that higher air temperatures tend to coincide with increased wind speeds, consistent with mesoscale circulation patterns at mid-latitudes and equatorial regions.

Conversely, `AIR_TEMP_C` and `GROUND_TEMP_C` exhibit moderate negative correlations with `LOCAL_RELATIVE_HUMIDITY` (-0.56 and -0.54, respectively). Although Martian relative humidity is extremely low compared to Earth, localized fluctuations do occur, notably in polar regions and during certain seasonal intervals. The inverse relationship between temperature and relative humidity mirrors terrestrial behaviour: warmer air retains proportionally less relative humidity when absolute water vapour content remains similar.

Additionally, `HORIZONTAL_WIND_SPEED` shows a moderate negative correlation with `LOCAL_RELATIVE_HUMIDITY` (-0.45) and a weak positive correlation with `RDS_T` (0.44). These associations may be explained by advective transport and surface heating processes associated with the movement of dry air masses, phenomena that are particularly relevant in regions subject to dust storms or strong topographic gradients.

In Figure 4.3 we present the distributions and boxplots for the dataset variables. Both air and surface temperatures exhibit right-skewed distributions, indicating that most observations lie in markedly negative ranges, which is consistent with the established mean conditions of the Martian surface. The occasional positive temperature readings, though infrequent, are likely attributable to strong diurnal variations driven by direct solar exposure and the surface's low thermal inertia. Atmospheric pressure, in contrast, displays an approximately symmetric distribution centred around 720 to 740 Pa, reflecting Mars' tenuous atmosphere with a density of less than 1% that of Earth's. This relative pressure stability suggests that the measurements were obtained under broadly consistent topographic and seasonal conditions.

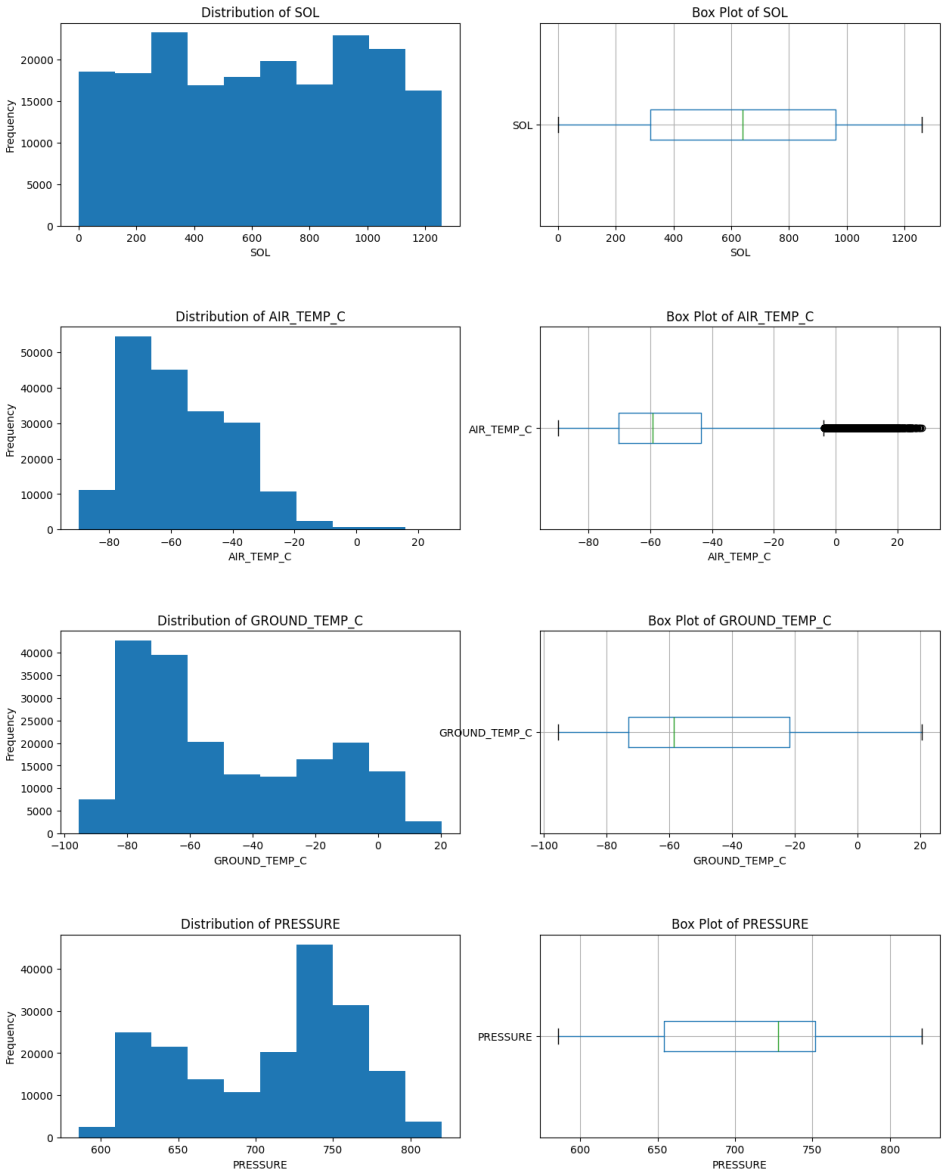

Figure 4.3: Histograms and Boxplots of SOL, `AIR_TEMP_C`, `GROUND_TEMP_C`, and PRESSURE

Continuing in Figure 4.4, the distributions reveal further insights into the dataset's non-Gaussian behaviour. The relative humidity variable exhibits a strongly right-skewed distribution with a pronounced concentration of values near zero, underscoring the arid character of the Martian atmosphere. Nonetheless, occasional higher humidity readings—treated as statistical outliers—suggest episodic events of localised condensation or transient increases in water vapour, potentially linked to surface ice sublimation or nocturnal deposition.

Horizontal wind speed likewise follows an asymmetric distribution, concentrated at low values (approximately 1–5 m/s), which is consistent with expectations given Mars' low atmospheric density. Higher recorded wind speeds appear as sporadic events and likely correspond to intense gusts or transient phenomena such as dust devils, which are frequently detected by optical and meteorological sensors.

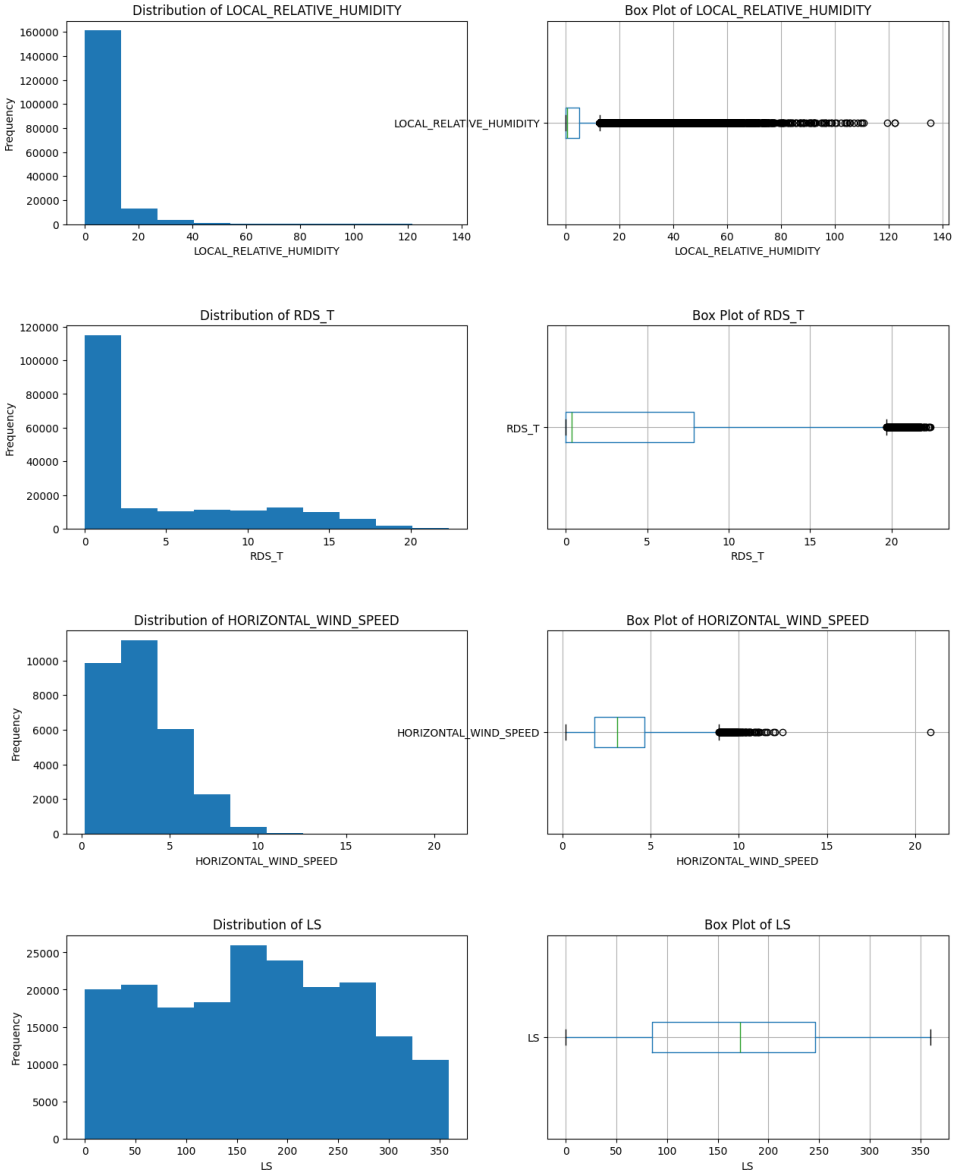

Figure 4.4: Histograms and Boxplots of SOL, `LOCAL RELATIVE HUMIDITY`, `RDS_T`, `HORIZONTAL WIND SPEED_T` and LS

Of particular interest is the behaviour of **RDS_T**, the total radiative flux measured by the RDS, which aggregates readings from multiple sub-sensors (e.g. *RDS_1*, *RDS_2*). The distribution of **RDS_T** is characterised by a high frequency of low values and a long right-hand tail of elevated measurements, indicating that incident radiation is often moderate or low—likely a consequence of persistent atmospheric dust. The radiation peaks (appearing as *outliers* in the boxplot) are plausibly associated with clear-sky intervals or seasonal events such as the Martian *perihelion*, when the planet is closest to the Sun and incoming solar flux attains maximum values. The pronounced skewness of **RDS_T** highlights substantial variability in illumination conditions on Mars, making this variable particularly relevant for studies of solar-panel performance, surface energy balance, and habitability.

Overall, the analysed data reflect the severe and highly variable nature of the Martian environment. The prevalent asymmetries across most variables emphasise the dominance

of extreme states and the need for careful detection of anomalous observations, which can have significant operational and scientific implications. Radiation, as quantified by **RDS_T**, emerges as a critical factor for both habitability assessments and the integrity of technological systems on Mars; analysing it jointly with other meteorological variables can reveal interaction patterns among dust, insolation, and atmospheric dynamics. The statistical analysis presented here contributes to a more robust understanding of Martian climate and to the refinement of environmental models used in planning future exploration missions.

4.1.2 Imputing missing data under 10%

Building on the observations from the previous section, we proceeded with the implementation of a predictive model to simulate Martian conditions. Due to the presence of missing values, however, the dataset required careful preprocessing to ensure the simulation would be realistic.

To address missing data, two principal strategies were applied: imputation and interpolation. For imputation we evaluated two distinct methods. First, the SimpleImputer, which replaces missing values using simple statistical rules (mean, median, or most frequent value). Second, a KNN imputer, which estimates missing entries by aggregating measurements from the closest neighbours in feature space. This combination allowed us to compare basic and more context-aware approaches for handling data absence.

Simple Imputer vs KNN Imputer

The SimpleImputer is a practical tool for handling missing values in datasets. It provides several replacement strategies—such as mean, median, or most-frequent value—and the choice of strategy should be guided by the data type and the analytic context.

Table 4.5: Pivot table of strategy configurations across environmental variables.

Strategy Config	AIR_TEMP_C	GROUND_TEMP_C	LOCAL_RELATIVE_HUMIDITY	PRESSURE	RDS_T
Constant					
False	3453.319185	3182.238085	92.335335	505424.897483	50.496905
True	3453.319185	3182.238085	92.335335	505424.897483	50.496905
Mean					
False	301.057775	849.904802	72.780684	3073.426756	32.580865
True	301.057775	849.904802	72.780684	3073.426756	32.580865
Median					
False	310.004913	957.429243	88.370433	3432.344312	47.396345
True	310.004913	957.429243	88.370433	3432.344312	47.396345
Most Frequent					
False	541.926354	1923.417973	92.335335	7709.839701	50.499531
True	541.926354	1923.417973	92.335335	7709.839701	50.499531

To fill missing values in the environmental variables we applied the SimpleImputer module and evaluated several imputation strategies. In the first stage we used mean imputation (strategy='mean'), replacing missing entries with each variable's mean. While this approach

centres the data around a representative value, it tended to reduce the original variability for skewed distributions.

Next, we applied median imputation (strategy='median'), which proved more robust to outliers and better preserved the distributional shape of variables with long tails. This strategy was particularly suitable for features exhibiting pronounced skewness, although it smoothed abrupt local transitions in some spatial regions.

We also tested mode imputation (strategy='most_frequent'), which is especially effective for discrete or categorical features such as wind orientation. However, mode imputation produced repeated values across extended areas, decreasing local heterogeneity.

Finally, constant-value imputation (strategy='constant') was evaluated: missing entries were replaced by a predefined constant. This method provided finer control over the imputation impact and was useful to mark unknown or unexplored regions without substantially altering the original data patterns.

Turning to the KNN Imputer:

The KNN imputer is an advanced, proximity-based method that uses the K-Nearest Neighbours algorithm to estimate missing values from features of the nearest observations. By exploiting similarity in feature space, the KNN approach provides a more context-aware and potentially more accurate imputation.

Concretely, the KNN imputer replaces a missing entry with the (typically weighted) average of the values of its k nearest neighbours, where k is an adjustable hyperparameter. Neighbourhoods are commonly defined using the Euclidean distance, although alternative distance metrics may be employed depending on the data characteristics and algorithm configuration.

Table 4.6 summarises the KNN imputer experiments conducted across different neighbourhood sizes (k = 2, 3, 4, 5, 7, 11, 15) and three neighbour-search algorithms: *auto*, *brute* and *ivfflat*. The results exhibit substantial variation in imputation accuracy depending on the chosen configuration. In general, smaller values of k combined with the *ivfflat* search produced the lowest mean absolute errors across all variables, indicating better adherence to local data structure.

For instance, *ivfflat* with k = 2 yielded markedly lower errors, particularly for air temperature (≈ 0.95), relative humidity (≈ 0.30) and pressure (≈ 0.08). By contrast, larger neighbourhoods (e.g. k = 11 or 15) tended to increase error: with *ivfflat* and k = 15 the MAE for air temperature exceeded 5.9, reflecting the loss of local specificity as more neighbours are included.

Across all tested values of k, the *auto* and *brute* search strategies delivered very similar performance, likely because both perform exact neighbour searches. The *ivfflat* method, which uses approximate search, proved more efficient in our experiments, especially for small k. This suggests that for datasets where local patterns are important for imputation, approximate neighbour-search methods can offer a favourable trade-off between accuracy and computational cost.

In summary, the KNN imputer's hyperparameters—most notably the number of neighbours and the search algorithm—substantially affect imputation quality. For datasets with pronounced local structure, adopting small k values together with lightweight approximate search algorithms such as *ivfflat* is recommended.

Table 4.6: KNN Imputer Performance in Different Configurations

Algorithm	AIR TEMP _C	GROUND _TEMP _C	LOCAL_RELATIVE HUMIDITY	PRESSURE	RDS_T
Auto					
auto_n11	5.040591	3.998186	1.886294	1.890220	0.262100
auto_n15	6.070184	3.751197	2.224172	2.028840	0.247970
auto_n2	3.234109	2.186726	0.852512	0.712185	0.097455
auto_n3	2.754043	2.111198	0.730058	0.559742	0.106265
auto_n4	2.392921	1.662882	0.652742	0.529638	0.131300
auto_n5	2.514762	1.868201	0.796934	0.571475	0.107769
auto_n7	2.830094	2.355307	1.046749	1.009967	0.145817
Brute					
brute_n11	5.040591	3.998186	1.886294	1.890220	0.262100
brute_n15	6.070184	3.751197	2.224172	2.028840	0.247970
brute_n2	3.234109	2.186726	0.852512	0.712185	0.097455
brute_n3	2.754043	2.111198	0.730058	0.559742	0.106265
brute_n4	2.392921	1.662882	0.652742	0.529638	0.131300
brute_n5	2.514762	1.868201	0.796934	0.571475	0.107769
brute_n7	2.830094	2.355307	1.046749	1.009967	0.145817
IVFFlat					
ivfflat_n11	4.595265	3.959454	1.670298	1.805002	0.256333
ivfflat_n15	5.933344	3.680331	2.277182	1.992149	0.246017
ivfflat_n2	0.953816	0.603522	0.300154	0.083241	0.029493
ivfflat_n3	1.121065	1.237252	0.383369	0.216152	0.078197
ivfflat_n4	1.306529	0.885863	0.450637	0.263015	0.103055
ivfflat_n5	1.814089	1.185088	0.643723	0.510092	0.095110
ivfflat_n7	2.698928	1.958164	0.995942	0.854773	0.141852

4.1.3 Imputing missing data above 10%

To handle missing values that exceed 10%, a different strategy was adopted because, given the volume of absent entries, the previously evaluated imputers were not viable for reliably filling those gaps. Since these variables are nevertheless important for the simulation, semi-supervised approaches were implemented: one machine-learning model and one deep-learning model, namely a *VAE* and a *Random Forest*.

The *VAE* is a generative model that learns a compact latent representation of the data and can generate new samples that resemble the original observations. It comprises two principal components: an encoder that maps input examples to a latent distribution, and a decoder that reconstructs the inputs from samples drawn from that distribution. Training minimises a combination of reconstruction loss (e.g. mean-squared error) and a regularisation term that forces the latent distribution to approximate a standard normal distribution (the Kullback–Leibler divergence). This structure allows the VAE to capture complex joint patterns in the data and to synthesise plausible values for missing entries.

The *Random Forest* is an ensemble learning method based on decision trees. It constructs a large number of trees during training and produces predictions by averaging the outputs of the individual trees. Random forests are relatively robust to overfitting, handle heterogeneous

data types well, and can accommodate missingness via surrogate splits or by operating on subsets of features. For these reasons, the algorithm is a common, practical choice for predictive imputation tasks.

Gradient Boosting VS Random Forest

This phase began with a preprocessing stage applied to the measurements collected by the *Perseverance* rover in the Jezero crater. The primary objective was to ensure that the data supplied to the semi-supervised learning algorithms had sufficient quality and structure for reliable training.

Continuous features were subsequently subjected to normalization to place variables on comparable scales and prevent attributes with large magnitudes from dominating model behaviour. This step improves numerical stability and promotes faster, more consistent convergence during optimization.

For the semi-supervised workflow, the dataset was partitioned into three disjoint subsets:

- **Labeled training set**: a small fraction of the data containing the known target values.
- **Unlabeled training set**: the majority of the observations, for which target values were masked to simulate limited supervision.
- **Test set**: held out for final evaluation to ensure that validation metrics reflect generalisation to unseen data.

Splits were performed in a stratified manner to preserve the marginal distribution of target variables across subsets. We also parametrised the proportion of labeled data available during training to assess how the amount of supervision affects model performance.

The data were organised into mini-batches suitable for semi-supervised training, allowing joint processing of labeled and unlabeled examples during each optimization step.

We employed a *Random Forest* regressor to predict horizontal wind speed from the available meteorological features and performed a targeted hyperparameter search to identify high-performing configurations. The search considered `n_estimators` in [10, 80], `max_features` in [0.3, 0.9], `min_samples_split` in [2, 10], `min_samples_leaf` in [1, 4], and different numbers of bins (`n_bins`) for discretising continuous variables: 8, 16, 32, 64, 128. We observed a consistent reduction in MSE when increasing both the number of bins and the number of trees, indicating that configurations with higher model capacity and finer feature discretisation tended to yield better predictive performance. The best results were obtained with 80 estimators, `max_features`=0.5, `min_samples_split`=5, `min_samples_leaf`=2 and `n_bins`=128, suggesting that the model is sensitive to parametrisation and benefits from careful tuning even in partially labeled scenarios.

The *VAE* model was developed to improve prediction of horizontal wind speed by leveraging all available meteorological variables. We implemented a compact architecture consisting of a single hidden layer in both the encoder and the decoder, each with 32 neurons and *Tanh* activations to promote numerical stability. The latent dimensionality was fixed at 16 to reduce overfitting. The loss combined a reconstruction term MSE with the KL divergence; however, the KL term was heavily down-weighted using $\beta = 0.0001$, yielding behaviour closer to a traditional autoencoder. An additional penalty was applied to the reconstruction error of the target variable to encourage the model to prioritise accurate prediction of wind speed. Training used a small learning rate (1×10^{-4}), a batch size of

64, and early stopping with a patience of 20 epochs. Despite these optimisation efforts, the VAE's predictive performance was unsatisfactory: it produced $R^2 = -0.1023$ and MAE = 0.1376 on the target variable. These results suggest that, although the model improved general reconstruction over training, it failed to capture the specific temporal and cross-variable patterns required for reliable prediction—possibly due to limitations in the latent representation or the intrinsic complexity of the data.

4.2 Training the Weather model of Mars

With missing values imputed, the next stage consisted of training predictive models to simulate Martian weather conditions. Recent literature on terrestrial weather forecasting (e.g. Reichstein et al. 2019; Habib et al. 2025) indicates that neural network architectures perform well on such tasks. Accordingly, we implemented and compared two deep models: LSTM, widely used for time-series forecasting, and N-BEATS, a more recent architecture that has shown competitive results in forecasting benchmarks. Both models were developed using the *PyTorch* framework and were configured to address the particularities of the Martian meteorological dataset.

Data preparation began with loading the imputed dataset and separating features into numerical and categorical types. Target variables—those the models are required to predict—were explicitly isolated from input features. Observations were ordered by the `SOL` column to preserve temporal coherence.

A temporal split was adopted: 90% of the data for training and validation and 10% held out for testing. This chronological holdout prevents leakage of future information into past observations, which is essential for valid time-series evaluation. Remaining missing values (if any) were handled using forward-fill followed by backward-fill to guarantee complete records for training.

This holdout approach refers to the practice of setting aside a portion of the dataset that is not exposed to the model during training, thereby serving as an unbiased benchmark to assess the model's generalization performance on unseen data.

Numerical features were standardised using `StandardScaler` to zero mean and unit variance; categorical features were encoded with one-hot encoding (dropping the first category to avoid collinearity). Crucially, all preprocessing transformers were fit only on the training split to avoid information leakage.

We then applied a sliding-window transformation to create supervised examples: each input sample comprised a lookback window of 72 time steps and the corresponding target(s) for the subsequent horizon. The resulting pairs were converted into PyTorch tensors and organised into *Dataset* and *DataLoader* objects for efficient batched training (train, validation and test).

This pipeline ensures statistically sound and computationally efficient data preparation, preserving essential temporal structure and providing a fair basis for comparing the LSTM and N-BEATS architectures.

4.2.1 N-BEATS

The *N-BEATS* model consists of a deep neural network architecture dedicated to univariate time series forecasting, as proposed by Oreshkin et al. 2019. The central proposition is

to demonstrate that densely connected layers, when organised in an appropriate residual topology, are capable of competing with, and even surpassing, traditional approaches based on statistical components such as ARIMA or Holt-Winters. Given a vector of historical observations $y_1, \ldots, y_T$, N-BEATS generates point forecasts for a horizon H, defined as:

$$\hat{y}_{T+1}, \hat{y}_{T+2}, \ldots, \hat{y}_{T+H} \tag{4.1}$$

without requiring any pre-specification of statistical model, manual feature extraction, or elaboration of time series-specific parameters.

Each processing unit, termed a *block*, receives as input a lookback vector x^ℓ and simultaneously produces an estimate of reconstruction of this same history (*backcast*) and a partial contribution to the future forecast (*forecast*), denoted respectively by:

$$\hat{x}^\ell \quad \text{(backcast)} \qquad \hat{y}^\ell \quad \text{(forecast)} \tag{4.2}$$

Internally, the block employs a sequence of fully-connected layers interspersed with ReLU activation functions to estimate expansion coefficients, which are then combined by basis layers — parameters that can be learned or pre-defined — resulting in the above vectors.

The architecture adopts a *double residual stacking*. At each block, the estimated backcast is subtracted from the input signal to ensure that subsequent blocks focus only on the unexplained components:

$$x^{\ell+1} = x^\ell - \hat{x}^\ell \tag{4.3}$$

The partial forecasts, in turn, are cumulatively summed to form the model's final forecast:

$$\hat{y} = \sum_\ell \hat{y}^\ell \tag{4.4}$$

This organisation favours gradient flow during training and endows the model with the capacity for progressive refinement of forecasts.

To provide interpretability to the process, the extended version of N-BEATS explicitly separates trend and seasonality components in two distinct *stacks*. The trend *stack* uses low-degree polynomial bases:

$$g_{\text{trend}}(\theta^f) = \sum_{i=0}^{d} \theta_i^f \, t^i \tag{4.5}$$

allowing the model to fit smooth curves over time. The seasonality *stack*, on the other hand, employs Fourier functions, modelling cyclic patterns through sines and cosines:

$$g_{\text{season}}(\theta^f) = \sum_{i=1}^{d} \theta_i^{(s)} \cos\left(\frac{2\pi i t}{H}\right) + \theta_i^{(c)} \sin\left(\frac{2\pi i t}{H}\right) \tag{4.6}$$

The output of the trend *stack* is removed before entering the seasonality *stack*, so that each component is estimated in isolation and, when combined, reproduce the global forecast. This decomposition reflects a direct analogy with classical time series methodologies, yet achieved through deep learning.

Additionally, N-BEATS achieves high levels of robustness and accuracy through an *ensemble* of model instances, differentiated by optimization metrics (such as sMAPE, MASE and MAPE), varied input window lengths (typically between $2H$ and $7H$), and multiple random initializations. The final aggregation of forecasts is done through the median of the individual results:

$$\hat{y}_{\text{final}} = \text{median}\left(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_M\right) \tag{4.7}$$

where M is the number of models in the *ensemble*. In the M3, M4, and TOURISM benchmarks, this strategy provided gains of up to 11% compared to the reference statistical methods. Thus, N-BEATS sets a new standard for univariate time series forecasting, combining architectural simplicity, application flexibility, and the possibility of interpreting classical components (Oreshkin et al. 2019).

Implementation of N-BEATS

In Figure 4.5 we have an architecture composed of several sequential blocks. Each block contains linear layers followed by ReLU and Dropout, and generates two outputs: the backcast, which is subtracted from the input to produce a residue, and the forecast, which contributes to the final prediction. The residue is passed to the next block, allowing each block to focus only on the part of the information not yet explained. In the end, all partial predictions are summed, resulting in the final output of the model.

After several trainings it was noticed that the version in figure 4.5 was not capturing the patterns well, so an enhanced version was developed. The version used in the project is a custom adaptation called *EnhancedNBeatsNet*, composed of multiple stacked blocks, each responsible for generating predictions and correcting the residues from the previous step. Each block (*EnhancedNBeatsBlock*) has a series of linear layers interleaved with ReLU activation functions and dropout layers, aiming to capture nonlinear patterns and prevent overfitting. Internally, each block performs two main operations: a forecast and a backcast, so that the residuals between the previous outputs and the backcasts are used as input for the following blocks. At the end of the network, the forecasts from all blocks are summed, forming the final output of the model, as shown in Figure 4.6.

Everything starts with the input vector, which contains the observations of a temporal window. This vector is forwarded to the first block, where it goes through a sequence of operations that combine linear transformations, normalization (LayerNorm), ReLU activation, and Dropout. This combination of operations allows the model to learn nonlinear relationships stably while reducing the risk of overfitting.

Each block generates two distinct results. The first is the backcast, which seeks to reconstruct the already known part of the time series. This reconstruction is used to update the residual, that is, the difference between the original input and the block reconstruction. The updated residual moves on to the next block, so that each block learns to deal only with information that previous blocks might not have captured. The second output is the forecast, which corresponds to a partial prediction for the future. Unlike the backcast, this does

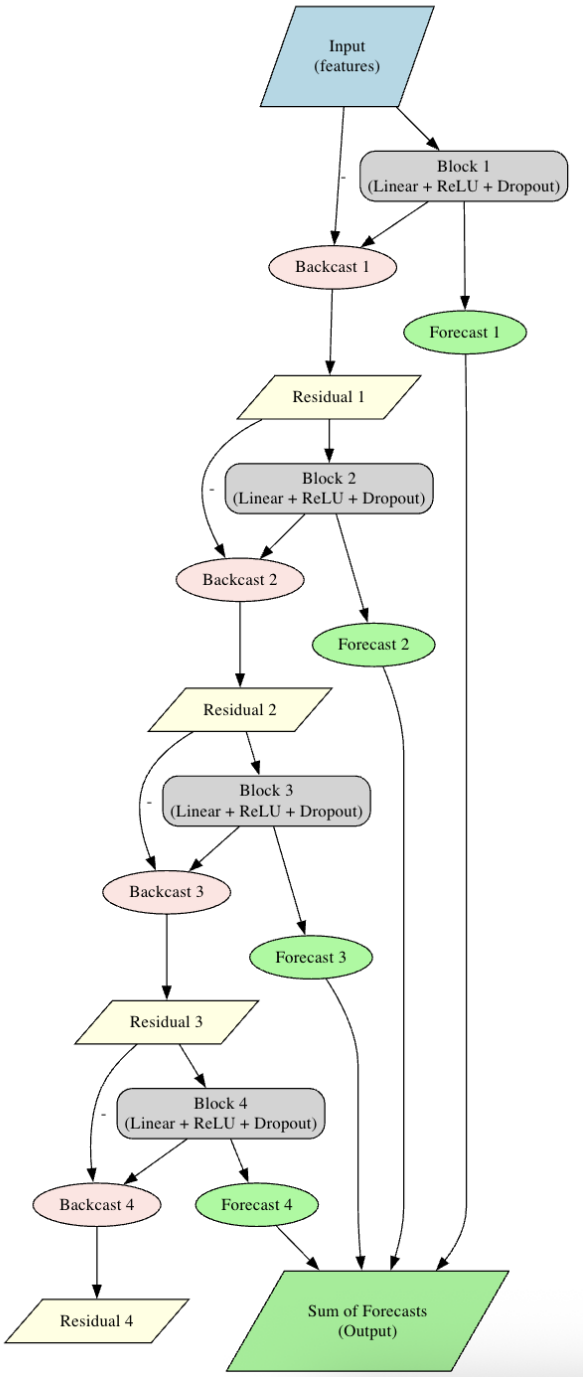

Figure 4.5: N-BEATS Architecture with (Linear, ReLU, and Dropout)

not interfere with the residual: all forecasts are accumulated and added together, resulting in the network's final forecast.

The vertical representation of the diagram helps to visualize this iterative process: from top to bottom, the information successively traverses the blocks, being refined at each step. In each of them, the backcast helps to reduce the error and the forecast adds a contribution to the global prediction. In the end, the sum of the individual forecasts generates an output

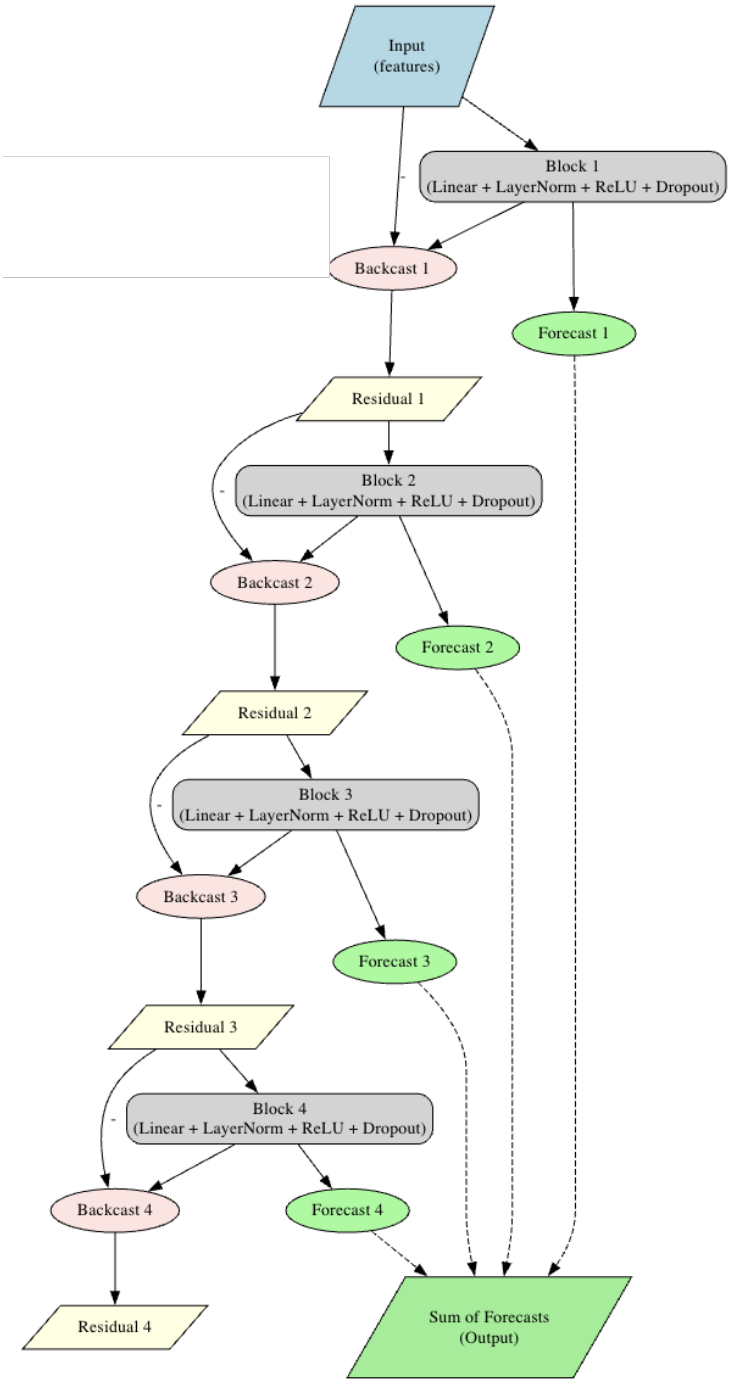

Figure 4.6: Enhanced N-BEATS Architecture with (Linear, LayerNorm, ReLU, and Dropout)

that combines the information from all the blocks.

The distinctive feature of this architecture lies precisely in this double mechanism: while the backcast ensures that the blocks do not merely repeat what has already been learned, the forecast allows each block to add its own perspective to the final prediction. In the Enhanced version, the introduction of Layer Normalization and Dropout reinforces the model's robustness, allowing it to generalize better in complex and highly variable contexts, as occurs in

environmental time series.

Thus, the diagram not only represents the passage of data through the network, but also illustrates the central philosophy of N-BEATS: a process of successive decomposition and reconstruction, where each block cooperatively contributes to a more accurate and informed final prediction. The hyperparameters used for N-BEATS were adjusted to the context of Martian simulation. The model receives as input temporal sequences of 120 steps (corresponding at 5m), and predicts the next 1 step. Each input is flattened to a one-dimensional vector, respecting the format required by the architecture. Ten blocks were defined (`num_blocks`=10), with three layers per block (`layers_per_block`=3) and a hidden dimension of 256 neurons. The model was trained for up to 1000 epochs, with an initial learning rate of 1×10^{-3} and L2 regularization via weight decay of 1×10^{-6}. To mitigate overfitting and accelerate convergence, *early stopping* was employed with a patience of 10 epochs, and adaptive learning rate reduction with `ReduceLROnPlateau`.

After changing the architecture, it was identified that the model retained several errors in the next prediction and biased the result in the long term. Based on the article Wilson 2016, it was seen that this is normal in autoregressive models, so a clipping was added that basically restricts the prediction from going to maximum or minimum numbers of the interval, but even so it did not solve the problem. Then, a retraining of the model was performed with Gaussian noise in the main variables, and it was noticed that there was no longer such a margin of error while also maintaining good accuracy regarding the main dataset.

It was retrained again because the model with a 24(2h) sequence window was not capturing the patterns well, so it was trained with 120(10h) and thus showed better patterns in all variables.

4.2.2 LSTM

The LSTM architecture, proposed by Hochreiter and Schmidhuber 1997, was developed to address a fundamental limitation of classical recurrent networks: the difficulty of learning long-term dependencies due to the vanishing and exploding gradient problem. In algorithms such as *Backpropagation Through Time* or *Real-Time Recurrent Learning*, the error signal propagated through time tends to decay exponentially, which compromises the learning of correlations between distant events in the sequence.

To mitigate this problem, LSTM introduces a new memory unit — called a *cell* — that maintains an internal state controlled by multiplicative gates that regulate the flow of information. The central principle is to allow the gradient to flow consistently through the structure known as the *Constant Error Carousel*, preventing it from vanishing over time.

The architecture of the memory cell is composed of three main gates:

- **Input gate**: controls the extent to which new values update the cell state.
- **Forget gate** (introduced later): regulates the proportion of the previous state that should be retained.
- **Output gate**: decides how much of the internal state is revealed as output.

Formally, the internal state of the cell c_t is updated as follows:

$$c_t = c_{t-1} + i_t \cdot g(x_t) \tag{4.8}$$

where i_t represents the activation of the input gate and $g(\cdot)$ is an activation function (e.g., hyperbolic tangent). The cell's output is then modulated by the output gate:

$$h_t = o_t \cdot h(c_t) \tag{4.9}$$

where o_t is the activation of the output gate and $h(\cdot)$ is another activation function (often the hyperbolic tangent).

Each gate i_t, o_t (and eventually the forget gate f_t) is calculated as:

$$i_t = \sigma(W_{ix}x_t + W_{ih}h_{t-1} + b_i) \tag{4.10}$$

$$o_t = \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o) \tag{4.11}$$

The presence of these gates allows the network to learn when to store information, when to erase it, and when to expose it, which has proven essential for tasks involving long temporal lags. By keeping the backpropagation of errors intact over time, even with intervals exceeding 1000 steps, LSTM overcomes critical limitations of previous methods.

Furthermore, experiments have shown that LSTM is capable of solving tasks with long temporal lags and noisy signals that were not solvable by standard recurrent networks. These tasks include synthetic problems like the *adding problem*, learning temporal grammars, and series with noise and nonlinear dependencies.

Therefore, LSTM has become a milestone in sequence modeling, being the foundation for modern advances in areas such as speech recognition, machine translation, and natural language modeling (Hochreiter and Schmidhuber 1997).

Implementation of LSTM

The implementation of the LSTM model was carried out using the PyTorch library, allowing a flexible and efficient construction of the network architecture. The model was encapsulated in a custom class, where the network structure is composed of a sequence of three stacked LSTM layers (`num_layers=3`) with `hidden_size=128`, and default `tanh` activation. The LSTM layers were configured with `batch_first=True`, ensuring that the batch dimension is first, facilitating compatibility with the tensors processed by the `DataLoader` used as we can see in Figure 4.7.

At the end of the recurrent layers, the outputs from the last temporal step of the sequence are extracted and passed through a fully connected linear layer (`nn.Linear`) that performs the final projection to the target variables. The number of units in this layer corresponds to the number of meteorological variables predicted, i.e., five simultaneous outputs per prediction time.

During training, the model receives as input three-dimensional tensors of shape (`batch_size, sequence_length, num_features`). The chosen loss function was the mean squared error (`MSELoss`), a standard metric for regression in time series, being sensitive to larger deviations. For optimization, the Adam optimizer was used, with a learning rate of `5e-4` and `weight_decay=1e-5`, configured to prevent overfitting. In addition, gradient clipping was applied with a limit of 1.0, which prevents the explosion of gradients along the LSTM layers.

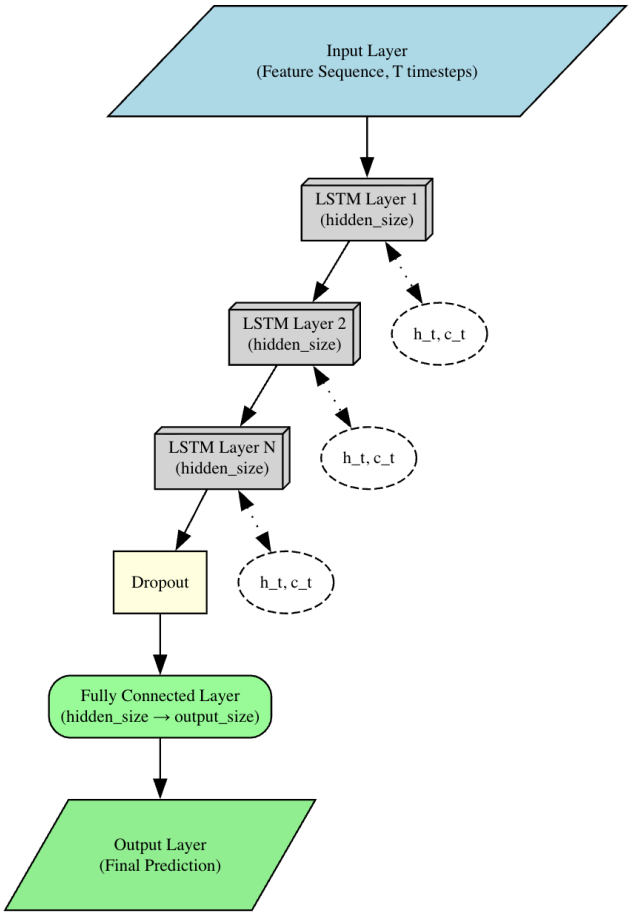

Figure 4.7: LSTM Architecture

4.3 Climate Managers Implementation

In addition to the meteorological variables implemented in Chapter 4.2, a subsystem was developed to generate extreme weather events. Examples include dust storms—common on Mars—and SEP events, which are solar eruptions capable of substantially altering the radiation environment at the Martian surface.

These events are simulated because they can disrupt rover operations, compromise scientific sampling activities, and pose risks to mission safety.

4.3.1 Solar Energetic Particles

SEP's are sporadic eruptive events that consist of energetic particles accelerated in the vicinity of the Sun. These storms are primarily composed of protons, but also include electrons and heavier ions, with energies ranging from a few keV to several hundred MeV, and occasionally even up to a few GeV. During these events, SEP fluxes can reach intensities significantly higher compared to Galactic Cosmic Rays, which are another omnipresent source of radiation in deep space. The frequency of SEP events is higher during solar maximum (Löwe et al. 2025).

Regarding rovers and space structures, spacecraft systems in Mars missions are also exposed to elevated radiation levels. The RAD, an instrument installed on the "Curiosity" rover of the MSL, was designed to measure the radiation environment, including SEP events, both in deep space and on the Martian surface. Since its launch, the MSL/RAD has detected 5 SEP events during the flight to Mars and 17 since its landing. Although the sources do not explicitly detail the specific damages caused to the electronic components of the rovers, the need to optimise shielding materials in spacecraft and space suits against radiation and the fact that systems are continuously exposed to high levels of radiation imply that radiation can adversely affect the functionality and longevity of unshielded equipment and structures (Löwe et al. 2025).

To mitigate these risks, forecasting and real-time monitoring of SEP events are crucial. Systems like the one developed for Mars missions can provide astronauts, and by extension, rover and structure operators, with enough time (at least 30 minutes) to avoid maximum radiation exposure and most of the cumulative dose of SEPs. This time allows astronauts to seek shelter in specially designed modules or underground bunkers on Mars. For rovers and other critical structures, the ability to quickly identify a SEP event is vital to implement safety procedures, such as shutting down sensitive equipment or repositioning shields, thus protecting these valuable assets (Löwe et al. 2025).

Implementation of SEP Manager

SEP events pose a significant environmental challenge for robotic mission simulations on Mars. To produce realistic scenarios, these events must be modelled consistently and with reasonable computational cost.

This work presents an algorithmic subsystem for simulating SEP events, conceptually inspired by the RDS_T channel of the MEDA instrument on NASA's Mars 2020 Perseverance rover. The implemented system is a configurable computational simulator that reproduces expected event behaviours using tunable parameters and probabilistic logic.

The architecture is organised into three modular components: (1) favourable-conditions detection, (2) event execution, and (3) temporal constraints enforcement. This modular separation facilitates maintenance and simplifies parameter calibration.

- $T_{cooldown} = 10$ SOLs (minimum configured period between events)
- $E_{max} = 3$ maximum events per 30-SOL period
- $D_{fixed} = 1$ SOL (default configured duration per event)
- $I_{SEP} = 0.6$ (configured radiation intensity)
- $P_{base} = 0.15$ (configurable base probability)
- RDS_T = simulated environmental radiation parameter (range: 0.0 to 1.0)

—

Algorithm 4.1 SEP Detection and Evaluation

Initialization:
$event\_active \leftarrow false$
$last\_end \leftarrow null$ ▷ Timestamp of the last event
$events\_in\_period \leftarrow 0$ ▷ Monthly/period counter
$period\_start \leftarrow 0$ ▷ Start of the current period

function CheckSEPEvent(RDS_T, Δt, $current\_step$)
    if $event\_active$ OR $RDS\_T < 0.001$ **then return** ▷ Avoid overlap or inappropriate ambient conditions
    end if
    Cooldown check:
    if $last\_end \neq null$ **then**
        $steps\_elapsed \leftarrow current\_step - last\_end$
        if $steps\_elapsed < T_{cooldown} \times 288$ **then return** ▷ Cooldown still active
        end if
    end if
    Frequency-limit check:
    if $current\_step - period\_start \geq 8640$ **then** ▷ Reset after 30 SOLs
        $events\_in\_period \leftarrow 0$
        $period\_start \leftarrow current\_step$
    end if
    if $events\_in\_period \geq E_{max}$ **then return** ▷ Monthly/period limit reached
    end if
    Probabilistic computation:
    $P_{hourly} \leftarrow P_{base} \times \frac{\Delta t}{3600}$
    $P_{adjusted} \leftarrow P_{hourly} \times (RDS\_T \times 2.0)$
    if $random(0, 1) < P_{adjusted}$ **then**
        TriggerSEPEvent($current\_step$)
    end if
end function

The algorithm in 4.1 implements the detection logic for candidate SEP events based on simulated radiation and temporal constraints.

Threshold system: The RDS_T threshold of 0.001 was established empirically during development to yield a realistic event frequency for the simulator. Values below this threshold are treated as insufficient ambient excitation.

Cooldown control: A configurable minimum interval between events is enforced (10 SOLs = 2880 simulation steps) to avoid unrealistically frequent consecutive events.

Probabilistic computation: The event probability is computed by scaling a base rate by elapsed time and amplifying it using the RDS_T channel; the factor 2.0 was selected empirically to calibrate system sensitivity.

—

Algorithm 4.2 SEP Event Execution and Control

```
1: function TriggerSEPEvent(current_step)
2:     if ValidateConstraints(current_step) = false then return ▷ Failed safety validations
3:     end if
4:     event_active ← true
5:     event_start ← current_step
6:     event_duration ← 295 ▷ Fixed duration: 1 SOL (in steps)
7:     events_in_period ← events_in_period + 1
8:     ActivateGlobalRadiation(I_SEP)
9:     Log("SEP event started")
10: end function
```

Algorithm 4.2 manages the life cycle of approved SEP events.

Event initialization: Coordinates activation of control flags, timestamps, counters, and the global radiation system with the configured intensity.

Temporal management: A timer monitors active events; when the configured duration (295 steps) elapses the finalization procedures are executed automatically.

—

Algorithm 4.3 SEP Constraints Validation

```
1: function ValidateConstraints(current_step)
2:     Cooldown check:
3:     if last_end ≠ null then
4:         steps_elapsed ← current_step − last_end
5:         if steps_elapsed < T_cooldown × 288 then
6:             return false ▷ Insufficient cooldown
7:         end if
8:     end if
9:     Frequency check:
10:    if events_in_period ≥ E_max then
11:        return false ▷ Period limit exceeded
12:    end if
13:    return true ▷ All constraints satisfied
14: end function
```

Algorithm 4.3 enforces the configured constraints that govern event frequency.

Cooldown enforcement: Verifies that the configured minimum period (10 SOLs) has elapsed since the last approved event; new events are rejected if this period is not respected.

Frequency enforcement: Applies the configured maximum of three events per 30 SOLs as a regulator, preventing excessively frequent events even under favourable conditions.

1. **Initial check**: Confirm $RDS\_T > 0.001$ and that no event is currently active.
2. **Constraints validation**: Verify minimum cooldown and period limit.
3. **Probabilistic computation**: Apply the configured probability formula.

4. **Stochastic decision**: Compare against a random draw.
5. **Execution**: If approved, start the event with the configured parameters.

The probabilistic formula implemented is:

$$P_{adjusted} = P_{base} \times \frac{\Delta t}{3600} \times (RDS\_T \times 2.0)$$

4.3.2 Dust Storms

Dust storms are among the most prominent and recurrent atmospheric phenomena on Mars. These storms arise from the interaction between wind and the Martian surface, where fine dust particles are lifted and transported through the thin atmosphere. The dust absorbs solar radiation and emits in the infrared, significantly altering the thermal structure and atmospheric dynamics (H. Wang and Richardson 2015).

Martian dust activity occurs across a wide range of scales, from local vortices (*dust devils*) to regional storms and even planet-wide dust storms capable of enveloping much of the globe for weeks or months. Classical studies classify storms with a major axis exceeding 2000 km as "non-local," encompassing both regional and planet-encircling events. Regional storms, in turn, are generally defined as events covering areas larger than 1.6×10^6 km^2 and persisting for more than three Martian days (H. Wang and Richardson 2015).

These storms have profound implications not only for the global climate of Mars but also for space missions, as they drastically reduce visibility and affect electronic components in probes and rovers. Furthermore, the aggregation of multiple local events can trigger large regional or even global storms, such as those recorded in 2001 and 2007 (H. Wang and Richardson 2015).

Thus, understanding the origin, evolution, and seasonal variability of these storms is essential for characterizing Martian climatology and predicting their impacts on future exploration missions.

Implementation of Dust Storm Manager

Understanding the atmospheric dynamics of Mars has revealed the fundamental role of variables such as **pressure**, **temperature**, and **LS** in identifying and characterizing dust storms. Measurements conducted by the *Perseverance* rover within Jezero Crater demonstrated that the pressure field undergoes daily and seasonal oscillations controlled by both thermal tides and the sublimation-condensation cycles of CO_2 at the poles. These variations not only reflect the planet's energy balance but also serve as precursors to dusty activity. During the passage of a regional storm at $Ls \approx 155°$, for instance, an abrupt drop of approximately 60 Pa in the daily minimum pressure was observed, accompanied by significant increases (up to fourfold) in the amplitude of the diurnal and semidiurnal components of thermal tides (Sánchez-Lavega et al. 2023).

Daily insolation, modulated by the planet's orbital position, acts as an additional trigger by making the atmosphere convectively unstable during the central hours of the day, favoring the formation of vortices and dust devils. Complementarily, the seasonal progression described by solar longitude exerts a decisive influence: starting at $Ls \approx 136°$, the increasing accumulation of suspended dust coincided with the intensification of pressure oscillations and baroclinic wave activity, often associated with the genesis of regional storms. Such evidence

confirms that the interaction between dynamic factors (pressure and temperature) and orbital factors (Ls) provides a robust framework for detecting and monitoring dust storms on Mars, as well as offering valuable insights for predictive modeling of the Martian climate.

Since dust storms are stochastic events, a probabilistic model was developed to estimate the likelihood of a storm based on current atmospheric conditions. The model considers three main variables: atmospheric pressure, air temperature, and LS. The probability is calculated using a logistic function that combines these variables, allowing dynamic adaptation to changing environmental conditions.

The criteria for initiating a storm are as follows:

1. `wind_speed_threshold` = 20.0 $\mathrm{m\,s^{-1}}$
2. `pressure_drop_rate` = 4.0 $\mathrm{hPa\,h^{-1}}$
3. `thermal_gradient` = 15.0 °C
4. seasonal Ls range = (200, 300)

Algorithm 4.4 Dust Storm Management

```
1: while Simulation Active do
2:     Update storm probability with update
3:     if Trigger conditions met (_check_trigger_conditions) then
4:         Initiate storm (_initiate_storm)
5:     end if
6:     if Storm Active then
7:         Update storm phase (_update_storm_phase)
8:         Apply effects on rovers (get_rover_effects)
9:     end if
10:    Update shared memory (_update_shared_memory)
11: end while
```

The **update** method is called periodically to ensure the continuous operation of the manager. During its execution, it checks the atmospheric conditions to determine if a storm should be initiated. To this end, the **_check_trigger_conditions** method evaluates whether the scientific thresholds have been reached. These thresholds include factors such as wind speed above 20 m/s, thermal gradient exceeding 15°C, and pressure drop rate greater than 4 hPa/h. Additionally, the system considers seasonality (LS between 200° and 300°) and applies a 72-hour cooldown after the end of a storm. If all conditions are met, the storm probability is calculated and compared with a random value. If the trigger is successful, the **_initiate_storm** method is called to start the storm.

Furthermore, the method updates the phases and intensity of the storm, applies the corresponding effects on the rovers, and keeps the shared memory updated with the latest data.

Dust storms follow a well-defined progression of phases. Initially, the **Calm** state indicates the absence of storms. When conditions are favorable, the storm enters the **Forming** phase, where it begins to develop. Subsequently, in the **Growing** phase, the storm's intensity gradually increases until it reaches the **Peak**, representing the moment of highest intensity.

After this peak, the storm enters the **Declining** phase, where its intensity starts to decrease, until it finally reaches the **Dissipating** phase, when the storm completely dissipates.

During storms, rovers are affected in several ways. Visibility is reduced, making navigation difficult. The speed of the rovers also decreases, and the LIDAR range is altered, impacting the ability to perceive the environment. These effects are calculated based on the intensity of the storm and are applied to simulate the adverse conditions faced by rovers on Mars.

4.4 Auction-Based Autonomy and Coordination Layer

This section presents the design and implementation of an auction-based coordination and autonomy framework in which explorer rovers discover samples and transporter rovers compete in auctions to collect them. The principal goal of this approach is to maximise the efficiency of sample retrieval by employing market-inspired mechanisms that enable decentralised, adaptive decision-making.

The motivation for this system lies in the operational constraints of planetary missions, where resources are scarce and operational failures carry high costs. Conventional exploration architectures that rely on direct, centralized control from Earth suffer from large communication delays and limited responsiveness. The hybrid architecture proposed here mitigates these issues by combining local autonomy with global coordination: explorer rovers operate autonomously to detect and characterise targets, while transporter rovers participate in competitive auctions to determine task assignment and execution.

The architecture is founded on distributed auction protocols: each collection task generated by an explorer is allocated through a competitive bidding process among available transporters. This mechanism not only improves assignment efficiency but also supports learning-enabled behaviours, allowing transporter agents to refine their bidding strategies over time. The framework is complemented by robust emergency protocols that preserve mission safety and continuity under adverse conditions (e.g., solar storms), ensuring that operational risk is managed while maintaining high scientific throughput.

4.4.1 Overall System Architecture

The implemented system follows a modular architecture composed of three primary, interconnected components: explorer rovers specialised in discovery, a centralized Communication Center responsible for coordinating operations, and transporter rovers tasked with sample retrieval. This design seeks to maximise operational robustness while maintaining the flexibility required to adapt to the dynamic conditions of the Martian environment as shown in Figure 4.8.

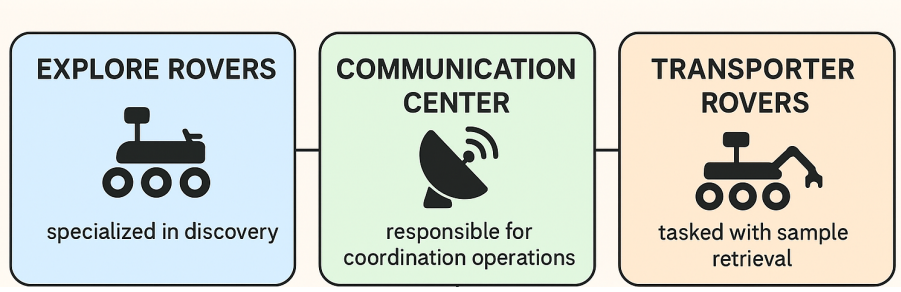

Figure 4.8: Diagram of overall system architecture

All rovers share the same discrete motion model: the eight cardinal and intercardinal directions (N, NE, E, SE, S, SW, W, NW), and the capability to move one cell per step, as illustrated in Figure 4.9.

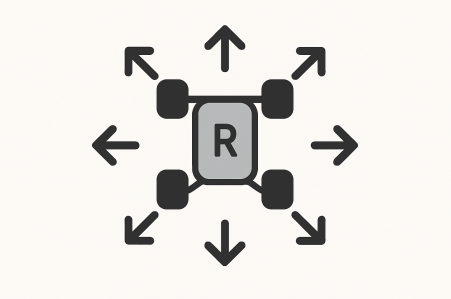

Figure 4.9: Possible rover movements

Explorer Agents

Explorer rovers constitute the primary discovery layer of the system, acting as intelligent mobile sensors that continuously expand the collective knowledge of the Martian terrain. Each explorer is designed as a fully autonomous agent, capable of making independent navigation decisions while contributing to the mission's global map.

The explorers integrate multiple specialised capabilities that operate in concert to maximise discovery efficiency. The detection and mapping subsystem is the most critical component: it enables safe navigation and the identification and cataloguing of scientifically relevant resources for subsequent retrieval.

To support the exploration strategies described above, each explorer embeds a compact short-range remote-sensing module. This module provides local perception—obstacle detection, terrain classification and the identification of potential scientific targets—within a circular neighbourhood of three-cell radius. The following subsections detail the implementation of the simulated LIDAR, the adopted ray-casting method, and the incremental local-map construction algorithm.

The detection system implemented on each explorer is an adaptation of the LIDAR concept for planetary environments. It allows a rover to "see" its surroundings within a three-cell radius, providing detailed information about obstacles, navigable terrain and potential scientific resources.

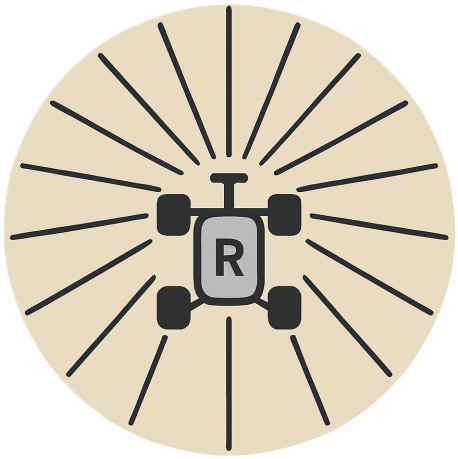

Figure 4.10: LIDAR Sensor Geometry example

The LIDAR system is implemented using a ray-casting algorithm in sixteen discrete directions, simulating laser beams that sample the surrounding terrain with high angular resolution. This approach provides near-complete coverage of the local neighbourhood while remaining computationally efficient for resource-constrained embedded platforms. Figure 4.10 illustrates the sensor geometry: the circle denotes the LIDAR range and the spokes represent the emitted rays in the sampled directions.

Algorithm 4.5 LIDAR Exploration Routine

```
position ← current position of the explorer
range ← 3                                                    ▷ cells
new_cells ← ∅
for dx ∈ [−range, range] do
    for dy ∈ [−range, range] do
        if dx² + dy² ≤ range² then
            new_position ← position + (dx, dy)
            if new_position is inside grid bounds then
                new_cells ← new_cells ∪ {new_position}
                Inspect terrain type at new_position
                Detect and classify samples and obstacles
            end if
        end if
    end for
end for
Transmit discovered cells and detections to the Communication Center
```

The effectiveness of this sensing subsystem derives from its incremental discovery capability. As each explorer traverses the terrain, the LIDAR module continuously expands the shared map, revealing previously unseen areas of interest and identifying obstacles that other rovers should avoid. This incremental strategy ensures systematic growth of the collective terrain representation and reduces redundant coverage by multiple explorers.

Complementing local sensing, continuous environmental monitoring is essential for mission safety and resilience. The RDS fulfils this role by correlating radiometric measurements with

trend estimates and operational rules to produce graded warnings. The following section details the RDS design, the filtering and trend-estimation methods used, and the criteria employed to escalate automated responses so that scientific assets and operational continuity are preserved, its operation is described in the Algorithm 4.5.

The RDS operates through four discrete alert levels, each corresponding to progressively severe radiation conditions and associated response protocols. The NORMAL level denotes nominal operating conditions in which all systems function without restriction. The WARNING level indicates elevated radiation that necessitates increased monitoring but does not prohibit routine operations. The CRITICAL level signals dangerous conditions that require preparations for evacuation, while the STORM_ALERT level triggers immediate evacuation procedures to pre-designated safe zones.

Algorithm 4.6 RDS Monitoring Routine

```
rds_history ← recent readings list
current_level ← read radiation sensor
rds_base ← −0.000309831                          ▷ baseline calibration constant
trend ← compute storm trend estimate
rds_t ← rds_base + trend + gaussian_noise
rds_history.append(rds_t)
alert_level ← determine level using configured thresholds
if alert_level ≠ previous_level then
    Send alert to Communication Center
    if alert_level ∈ {CRITICAL, STORM_ALERT} then
        Initiate evacuation procedures
    end if
end if
Update monitoring history and logs
```

The sophistication of the RDS lies in its predictive detection capability. Rather than merely reacting to hazardous radiation levels, the system analyses trends in historical measurements to anticipate solar particle events, enabling precautionary evacuation before conditions become critical. Such a proactive policy is essential in space environments where response delays can have severe consequences, so everything that has been described is present in the algorithm 4.6.

Integration between environmental monitoring and navigation policies is crucial to balance scientific discovery with operational safety. Accordingly, exploration strategies were designed to respond dynamically to signals from the RDS subsystem and other environmental variables: when the RDS indicates favourable conditions, explorers prioritise high-coverage, efficiency-oriented search modes; under elevated alerts, they switch to conservative behaviours that mitigate risk (for example, prioritising shadowed areas or awaiting the end of the event). This coordination yields adaptive exploration—efficient for prospecting new areas and robust in the face of adverse events—without requiring manual mode switching.

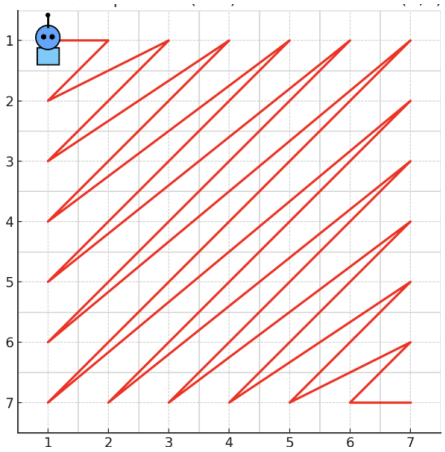

Figure 4.11: Frontier Expansion movement pattern of an explorer

The Frontier Expansion strategy focuses on extending the boundary of known terrain by prioritising movement to cells adjacent to already-explored areas. This approach ensures continuous growth of the collective map and minimises the risk of leaving coverage gaps, as illustrated in Figure 4.11.

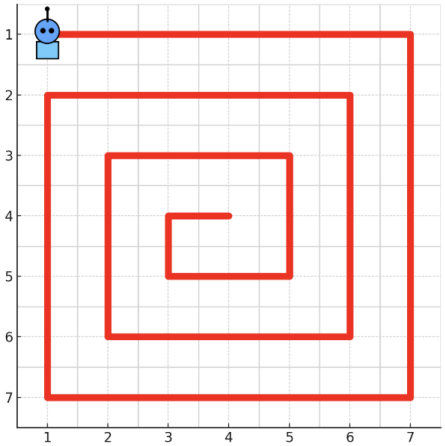

Figure 4.12: Spiral movement pattern of an explorer

The Spiral Patterns strategy implements a systematic spiral traversal that ensures uniform coverage of extensive areas. This method is particularly effective in open terrain with few obstacles that would interrupt regular movement. The spiral algorithm progressively expands the exploration radius, guaranteeing that each cell in the region is visited exactly once, as illustrated in Figure 4.12.

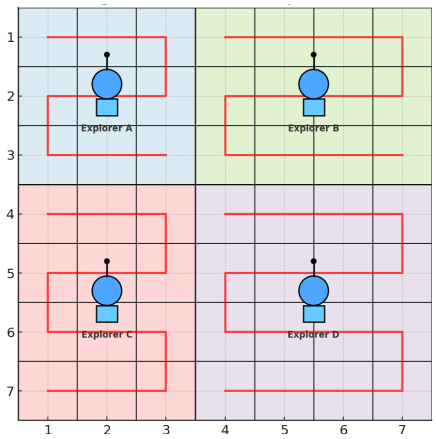

Figure 4.13: Grid Systematic Assignment movement pattern of an explorer

The Grid Systematic Assignment strategy partitions the Martian terrain into distinct sectors and assigns different explorers to individual regions using a systematic grid-scan approach. This scheme prevents unnecessary overlap between rovers and maximises collective exploration efficiency. The sector-assignment mechanism is dynamic, allowing reassignment of sectors when explorers complete their designated regions or when obstacles impede progress, as exemplified in Figure 4.13.

These strategies are formalised as shown in Algorithm 4.7.

Algorithm 4.7 Exploration Strategy Selection

$current\_position \leftarrow$ position of the explorer
$known\_terrain \leftarrow$ set of already explored cells
$frontiers \leftarrow$ identify adjacent unexplored cells
$known\_obstacles \leftarrow$ set of discovered obstacles
if $|frontiers| > frontier\_threshold$ **then**
    $strategy \leftarrow$ FRONTIER_EXPANSION
    $destination \leftarrow$ select nearest frontier
else if $obstacle\_density < spiral\_threshold$ **then**
    $strategy \leftarrow$ SPIRAL_PATTERN
    $destination \leftarrow$ next position in spiral pattern
else
    $strategy \leftarrow$ GRID_SYSTEMATIC
    $destination \leftarrow$ next cell in assigned sector
end if
Navigate to $destination$

To ensure these strategies can interoperate with other rovers and system components, a dedicated communication subsystem was designed to provide efficient sharing of critical information with the *Communication Center* and peer agents. Every significant discovery — whether a geological sample, a hazardous obstacle, or a shadow zone that can provide radiation shelter — is immediately reported via structured messages that carry the necessary metadata for coordinated response.

The communication protocol implements several message types, each optimised for specific operational purposes. Exploration update messages convey expansions of the shared map; sample-discovery messages announce valuable resources and trigger collection tasks; and RDS alert messages coordinate emergency responses to hazardous radiation conditions. This structured messaging framework ensures rapid dissemination of critical information while protecting the communication channel from saturation.

CommunicationCenter

The CommunicationCenter functions as the system's central coordination component, acting as an intelligent hub that aggregates information from all rovers and orchestrates their activities. It was designed to address the fundamental challenge of maintaining global coordination in distributed systems, ensuring that local decisions made by individual rovers contribute efficiently to the mission's collective objectives.

The CommunicationCenter's architecture adopts a single source of truth for the mission state, reconciling centralized information management with local autonomy. In this scheme the CommunicationCenter maintains and synchronises maps, inventories, positions and operational logs, while lightweight communication protocols and fault-tolerant policies enable agents to continue operating in degraded mode when connectivity is intermittent. The following section details the state-representation mechanisms, message formats and synchronization protocols that support global coherence and underpin the negotiation and task-allocation algorithms presented later.

The CommunicationCenter maintains a complete, up-to-date representation of mission state, including a detailed map of explored terrain, precise locations of detected obstacles, a comprehensive inventory of identified samples and their current status, and continuous monitoring of the positions and operational states of all active rovers.

State management is implemented using optimised data structures that permit efficient updates and fast queries. The explored-terrain map employs spatial indices to rapidly identify unexplored areas and to compute routes efficiently between locations. The sample inventory stores not only locations but also metadata such as sample type, estimated scientific value and the history of collection attempts.

As shown in Algorithm 4.8, the CommunicationCenter processes messages received from explorer and transporter rovers to update the global state. Each message is categorised by type and triggers specific actions based on its content: exploration updates mark newly observed cells on the global map; sample-discovery messages add entries to the inventory and create collection tasks; and obstacle reports update the obstacle map and adjust navigation routes as required.

Algorithm 4.8 Global State Update

```
1: procedure UpdateState(message)
2:     type ← message.type
3:     content ← message.content
4:     sender ← message.sender
5:     if type = "exploration" then
6:         for each cell in content.new_cells do
7:             global_map.mark_explored(cell)
8:         end for
9:         Update exploration statistics for sender
10:    else if type = "sample_discovery" then
11:        inventory.add(content.sample)
12:        Create a collection task for content.sample
13:        Notify available transporters
14:    else if type = "obstacle_detected" then
15:        obstacle_map.add(content.position)
16:        Update affected navigation routes
17:    end if
18:    Log the message in history
19: end procedure
```

Notably, the CommunicationCenter also implements a structured asynchronous messaging subsystem that enables rovers to transmit discoveries and state updates without interrupting their primary operations. This messaging layer is critical for preserving effective coordination when agents are geographically dispersed and subject to variable communication delays. Figure 4.14 illustrates the messaging architecture.

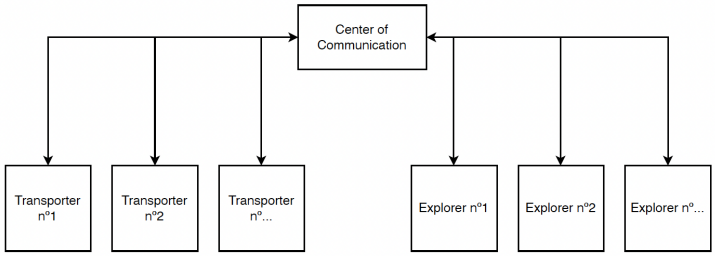

Figure 4.14: Structured Asynchronous Messaging System

The messaging framework supports multiple message types, each optimised for a specific operational purpose: exploration-update messages report expansions of the shared map; sample-discovery messages trigger automatic creation of collection tasks; RDS anomaly alerts coordinate emergency responses; and shadow-zone discovery messages convey critical sheltering information during solar storms.

The messaging subsystem's robustness is ensured through redundancy and integrity-checking mechanisms. Critical messages require explicit acknowledgement, and automatic retransmission is performed for messages unacknowledged within configured timeouts. These provisions

minimise the risk of losing essential information under degraded or intermittent communication conditions.

As an introductory coordination mechanism for the multi-agent system, the following section describes a deterministic, heuristic-based task-assignment framework. This baseline mechanism serves a dual purpose: it provides a well-understood reference for the evaluation of more advanced approaches, and it acts as a reliable fallback when learned policies are unavailable or produce unstable behaviour.

The baseline algorithm is intended as a proof of concept and a foundation for subsequent developments, exposing the core principles of multi-agent coordination prior to the integration of learning capabilities. The baseline was later extended to incorporate MARL techniques while remaining available as a fail-safe to preserve operational robustness.

Assignment decisions proceed in multiple phases. Transporter rovers are first categorised according to current workload: idle agents (no active tasks) receive highest priority for new assignments; busy agents (one to two active tasks) are considered for additional tasks only if no idle agents are available; overloaded agents (three or more active tasks) are excluded from new assignments until their load decreases. This process is formalised in Algorithm 4.9.

Algorithm 4.9 Deterministic Baseline Task Assignment

```
1:  procedure AssignTasksBaseline(transporters)
2:      free, busy, overloaded ← categorize_by_load(transporters)
3:      tasks_sorted ← sort_by_priority_and_proximity()
4:      assignments ← ∅
5:      for each task in tasks_sorted do
6:          candidates ← free
7:          if |free| = 0 then
8:              candidates ← busy
9:          end if
10:         best_rover ← null
11:         best_score ← −∞
12:         for each rover in candidates do
13:             score ← evaluate_deterministic_suitability(rover, task)
14:             if score > best_score then
15:                 best_score ← score
16:                 best_rover ← rover
17:             end if
18:         end for
19:         if best_score > minimum_threshold then
20:             assignments[task] ← best_rover
21:             Update_rover_category(best_rover)                ▷ move between
    free/busy/overloaded
22:         end if
23:     end for
24:     Execute_load_redistribution_if_necessary()
25:     return assignments
26: end procedure
```

The deterministic suitability assessment combines multiple predefined criteria — notably geographic distance to the sample, the transporter's current workload, and its historical performance on analogous tasks. This multi-criteria baseline algorithm yields consistent, predictable assignments grounded in well-established operational heuristics.

While effective in controlled scenarios, the deterministic baseline exhibits clear limitations in highly dynamic and uncertain environments such as the Martian surface. To address these shortcomings, the architecture was extended to include learning-based capabilities via MARL, while preserving the deterministic baseline as a robust operational fallback.

This extension represents a shift from fixed rule-based coordination to adaptive, experience-driven coordination: agents progressively refine their decision policies through operational feedback, thereby improving collective performance as they accumulate interaction data in the simulated Martian environment.

To prevent load imbalances in which some transporters become overloaded while others remain underutilised, the CommunicationCenter implements a dynamic load-balancing mechanism. The system continuously monitors task distribution across all transporters and triggers automated redistribution when significant imbalances are detected.

The redistribution routine evaluates workload metrics — including the number of active tasks, samples currently carried, and estimated remaining time for in-progress tasks — and, when inter-agent load differentials exceed predefined thresholds, reassigns tasks to restore a balanced collective workload.

Algorithm 4.10 Dynamic Load Balancing

```
1:  procedure BalanceLoad(assignments, transporters)
2:      loads ← compute workload for each transporter
3:      load_min ← min(loads)
4:      load_max ← max(loads)
5:      if load_max − load_min > threshold then
6:          overloaded ← transporter with maximum load
7:          underutilised ← transporter with minimum load
8:          transferable_tasks ← identify tasks eligible for transfer from overloaded
9:          for each task in transferable_tasks do
10:             transfer_cost ← estimate redistribution impact for task
11:             if transfer_cost < expected benefit then
12:                 Transfer task from overloaded to underutilised
13:                 Recompute affected routes and schedules
14:                 Notify involved transporters of the change
15:             end if
16:         end for
17:     end if
18: end procedure
```

Transporter Agents

Transporter rovers constitute the system's executive layer, responsible for converting discoveries into tangible scientific outcomes by collecting and delivering samples. These agents are designed as highly specialised platforms, equipped with advanced navigation subsystems,

manipulator hardware for sample acquisition, and safety protocols to operate reliably under adverse conditions as shown in Algorithm 4.10.

The transporter design integrates several interdependent modules to enable efficient autonomous operation. The navigation subsystem is the critical element: it builds on an adaptive implementation of the A* algorithm, optimised for dynamic multi-agent environments through intelligent caching and automatic replanning.

The pathfinding implementation uses classical A* as its core but incorporates multiple optimisations to improve runtime performance and robustness in changing conditions. An edge-cost cache accelerates repeated computations for frequently traversed corridors, and an automatic replanning mechanism is triggered when new obstacles or conflicts are detected.

When the environment changes substantially—for instance, when new obstacles appear or other rovers obstruct planned routes—the system invalidates the relevant cache entries and executes a full replanning cycle to guarantee that transporters maintain valid, efficient paths to their objectives. This strategy preserves operational resilience in highly dynamic scenarios.

The adaptation also includes specific features for multi-agent coordination: peer rovers are modelled as temporary, potentially mobile obstacles with predictable motion patterns, and the planner supports multiple heuristics selectable at runtime (e.g. a SHORTEST heuristic prioritising minimum path length and a BALANCED heuristic that additionally accounts for congestion and risk), as formalised in Algorithm 4.11.

Algorithm 4.11 Adaptive Multi-Agent A* Pathfinding

```
1:  procedure PlanAdaptivePath(start, goal, peer_rovers)
2:      temporary_obstacles ← positions of peer_rovers
3:      permanent_obstacles ← known obstacles from the map
4:      all_obstacles ← permanent_obstacles ∪ temporary_obstacles
5:      if new obstacles detected since last planning then
6:          Invalidate edge-cost cache
7:          explored_cells ← fetch already explored cells
8:          Run A* using the selected heuristic considering all_obstacles
9:          Store updated edge costs in the intelligent cache
10:     else
11:         Use existing path if still valid
12:     end if
13:     next_step ← first position on the computed path return next_step
14: end procedure
```

The efficiency of the adaptive A* planner is maximised by an intelligent caching mechanism that stores frequently used routes and reuses previous computations whenever possible. This cache is particularly valuable for corridors between the central base and commonly visited sample locations, where multiple transporters benefit from precomputed edge costs and path fragments.

Each transporter rover is governed by a finite-state machine that manages the complete sample-collection lifecycle, from initial planning to final delivery at the base. The state machine enforces correct sequencing of task phases and ensures transitions occur only when the requisite conditions are satisfied.

The PLANNING state corresponds to the initial phase in which a rover receives a collection assignment and computes an optimal route to the target sample. During planning the rover considers not only geographic distance but also traffic congestion, prevailing radiation conditions, and the availability of alternative routes should the primary path become obstructed.

The COLLECTING state covers navigation to the sample site and the physical acquisition procedure. In this phase the rover must handle dynamic obstacles, coordinate with nearby peers, and adapt its route in response to updated information received from the CommunicationCenter.

The DELIVERING state denotes the return journey to base with the collected sample. This phase comprises return navigation as well as coordination with base systems to effect an efficient handover and to prepare the rover for subsequent tasks, it is illustrated in Algorithm 4.12.

Algorithm 4.12 Transporter State Machine

```
1: procedure UpdateBehavior(transporter)
2:     current_state ← transporter.state
3:     if current_state = PLANNING then
4:         if task assigned and route computed then
5:             transporter.state ← COLLECTING
6:             Initiate navigation to sample
7:         end if
8:     else if current_state = COLLECTING then
9:         if arrived at sample location then
10:            Execute sample collection
11:            transporter.state ← DELIVERING
12:            Compute return route to base
13:        else
14:            Continue navigation to sample
15:            Check whether route requires replanning
16:        end if
17:    else if current_state = DELIVERING then
18:        if arrived at base then
19:            Deliver sample to base
20:            Mark task as complete
21:            transporter.state ← IDLE
22:            Request new task
23:        else
24:            Continue navigation to base
25:        end if
26:    end if
27: end procedure
```

One critical capability of the transporter rovers is their ability to respond appropriately to emergency situations, in particular solar particle events that may jeopardise both rover safety and the integrity of transported samples. The implemented emergency-response system is

based on a centralized communication architecture in which explorer rovers act as distributed radiation sensors.

Under nominal conditions transporters execute their assigned tasks. Emergency detection begins when explorers continuously monitor local RDS readings; upon detecting an anomalous radiation level an explorer immediately reports the observation to the *Communication Center* via a `rds_anomaly` message. The *Communication Center* applies a cross-validation policy, requiring confirmation from at least two independent explorers before declaring an active solar event.

When a solar event is confirmed, the *Communication Center* broadcasts a storm _status _update to all transporters. Transporters poll for these messages periodically using _check _storm _messages() and activate evacuation protocols immediately upon receipt of an active-storm confirmation.

Evacuation behaviour is multimodal and context dependent: a transporter at base on alert remains sheltered until conditions return to safe levels; a transporter executing a collection or delivery suspends the current task, persists the full task state in `suspended_task`, and navigates to the nearest shadow zone using the adaptive A* pathfinder to minimise radiation exposure, this system response emergency is illustrated in Algorithm 4.13.

Algorithm 4.13 Emergency Response Routine

```
1: procedure ManageEmergency(transporter, storm_messages)
2:     current_position ← transporter.position
3:     storm_status_changed ← check storm messages
4:     if storm_status_changed and storm active then
5:         if in safe zone or at base then
6:             evacuation_state ← "in_shelter"
7:             Remain at current position until the storm subsides
8:         else if evacuation_state ≠ "moving_to_shelter" then
9:             if has active task and not yet suspended then
10:                suspended_task ← persist the full task state
11:                Include: task, stage, path, collected_samples
12:            end if
13:            evacuation_target ← find nearest safe zone
14:            if evacuation_target found then
15:                evacuation_state ← "moving_to_shelter"
16:                Use A* to navigate to the safe zone
17:            else
18:                evacuation_state ← "stuck_in_storm"
19:                Remain stationary
20:            end if
21:        else
22:            Continue movement towards the safe zone using A*
23:        end if
24:    end if
25:    if storm_status_changed and storm inactive and has suspended task then
26:        Restore task from suspended_task
27:        Restore: task, stage, path, path_index, samples
28:        evacuation_state ← "none"
29:        Clear evacuation metadata
30:        Recompute route if necessary
31:    end if
32: end procedure
```

The system carefully preserves the state of all tasks suspended during emergencies, ensuring that once normal conditions are restored rovers can resume their activities precisely where they were interrupted. This state-preservation capability is essential to maintain operational efficiency during and after emergency events.

4.4.2 MARL System Architecture

The natural evolution of the deterministic baseline task-assignment system culminates in the integration of learning capabilities through MARL. This enhancement represents a substantive improvement over the traditional rule-based approach, allowing each rover to develop individually optimised strategies from operational experience and producing collective performance that improves continuously over time.

The MARL layer is fully compatible with the baseline: it functions as an intelligent overlay that replaces fixed heuristics with learned, adaptive policies. This hybrid architecture

preserves operational robustness—when MARL capabilities are available the system employs learning-enabled auction mechanisms; if they are not, it automatically reverts to the deterministic baseline to ensure uninterrupted mission operations.

MARL Coordinator

The MARLCoordinator represents the evolution of the traditional CommunicationCenter, extending its orchestration capabilities by incorporating learning-based auction mechanisms. This component replaces the deterministic baseline algorithm with a competitive auctioning framework in which each Transporter develops customised strategies grounded in accumulated operational experience.

The transition from the baseline to MARL preserves the system's core objectives — efficient task allocation and balanced workload distribution — while introducing adaptive capabilities that enable continuous optimisation driven by real operational feedback. This example illustrates how a deterministic module can be transformed into an intelligent component and demonstrates that MARL constitutes a direct evolutionary extension of the baseline system.

The MARL auction mechanism is a direct extension of the deterministic baseline, substituting fixed suitability evaluations with a dynamic, competitive process. Where the baseline relied on predefined heuristics for task assignment, the MARL framework allows each Transporter to learn and refine its own evaluation strategy based on historical performance and contextual experience.

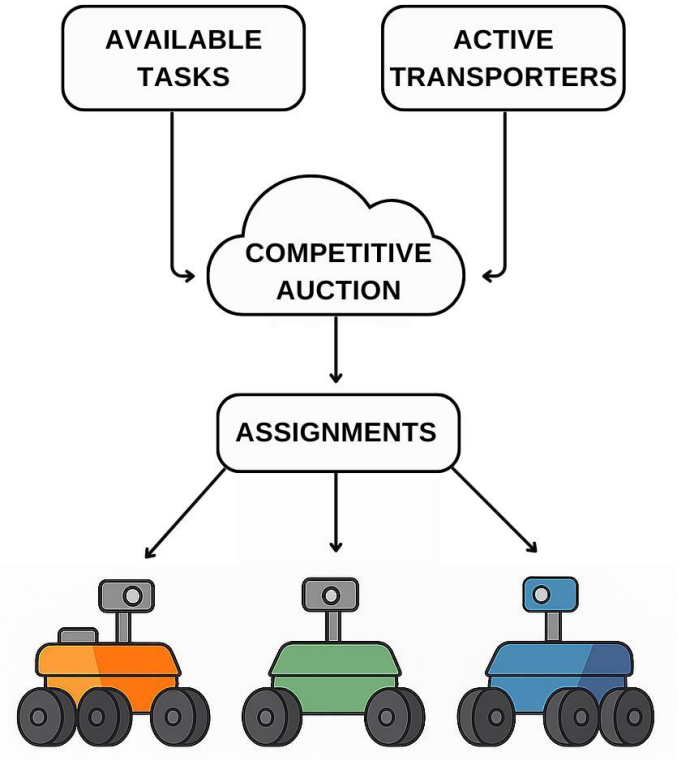

Figure 4.15: MARL Auction System

Each collection task triggers a competitive auction among all available Transporters, in which each Transporter submits a bid that reflects not only its technical capability but also a learned

confidence derived from prior experience with similar tasks. This mechanism constitutes a substantive improvement over the deterministic baseline, enabling continuous adaptation to dynamic operational conditions; the dynamics of the process are illustrated in Figure 4.15.

The auction maintains the baseline's organisational flow — identification of eligible Transporters, suitability evaluation, and selection of the preferred candidate — but replaces fixed suitability metrics with evolutionary bidding strategies that are refined over time through operational feedback.

Algorithm 4.14 MARL Auction System

```
1:  procedure ConductAuction(available_tasks, active_transporters)
2:      assignments ← ∅
3:      assigned_agents ← ∅
4:      for each task in available_tasks do
5:          eligible ← filter transporters with no active tasks
6:          eligible ← eligible \ assigned_agents
7:          if |eligible| = 0 then
8:              Stop auction — all agents are busy
9:          end if
10:         bids ← ∅
11:         for each rover in eligible do
12:             policy ← retrieve individual policy of rover
13:             bid ← policy.calculate_bid(task, competition_history)
14:             bids[rover] ← bid
15:         end for
16:         if |bids| > 0 then
17:             winner ← arg max_rover(bids[rover])
18:             assignments[task] ← winner
19:             assigned_agents ← assigned_agents ∪ {winner}
20:             Log auction in history for learning
21:             Notify all participants of the result
22:         end if
23:     end for
24:     return assignments
25: end procedure
```

Crucially, the system notifies all auction participants of the outcome, not only the winner. This information is consumed by each Transporter to refine future bidding strategies, enabling agents to learn competitive patterns and to adjust bid aggressiveness according to their historical successes and failures, we can better understand this flow in the algorithm 4.14.

After each task is completed (successfully or otherwise), the system collects detailed feedback on the responsible Transporter's performance. Collected metrics include completion time, route efficiency, resources consumed, and any complications encountered during execution. These observations are used to update the Transporter's policy, thereby improving its capability to evaluate and prioritise future tasks.

Learning is driven by a modular reward algorithm that aggregates multiple performance factors. Successful executions are rewarded according to completion efficiency, while failures

are penalised proportionally to their impact on the mission. The reward computation also incorporates contextual variables — for example, environmental conditions present during task execution — allowing Transporters to calibrate their expectations and behaviour to situational specifics. See Algorithm 4.15.

Algorithm 4.15 Continuous Learning System

```
1:  procedure UpdateLearning(rover_id, completed_task, result)
2:      policy ← retrieve policy for rover_id
3:      success ← result.was_successful
4:      completion_time ← result.time_spent
5:      difficulty ← completed_task.difficulty_level
6:      reward ← compute reward from multiple performance factors
7:      if success then
8:          reward ← reward× success_bonus_factor
9:          reward ← reward / max(1, completion_time/expected_time)
10:     else
11:         reward ← reward× failure_penalty_factor
12:     end if
13:     policy.apply_learning(reward, task_context)
14:     Update historical statistics for rover_id
15:     Persist updated policy
16:     Update global MARL performance metrics
17: end procedure
```

Adaptive Individual Policies

Each Transporter develops a distinct operational "personality" via an adaptive policy framework that evolves from individual experience. This design enables Transporters with similar physical capabilities to adopt different operational strategies, producing behavioural diversity that strengthens the collective robustness of the system.

An individual Transporter's policy is encoded as a small set of adaptive weights that modulate how the agent evaluates different task attributes. These weights are randomly initialised to ensure early diversity and are progressively refined through operational feedback and learning.

And how do these weights work?

Thus, each Transporter maintains five principal weights that define its operational personality. The distance-preference weight determines how the Transporter balances geographic distance against other factors — some Transporters may develop a bias for proximate tasks to maximise efficiency, while others may exhibit greater tolerance for long-range assignments. The load-tolerance weight governs how the Transporter evaluates additional tasks when already engaged — more confident Transporters may accept multiple concurrent assignments, whereas more conservative agents prefer to focus on a single task at a time.

The confidence-bias weight influences how aggressively the Transporter competes in auctions, with higher confidence producing larger bids and lower confidence yielding more cautious behaviour. The efficiency-focus weight controls the trade-off between rapid completion and thoroughness, and the risk-tolerance weight modulates the Transporter's willingness to accept tasks in potentially hazardous environments.

Beyond these core personality weights, each Transporter evolves a sophisticated bidding strategy driven by feedback from previous auctions. This strategy incorporates competitive awareness: Transporters learn to adjust their bids according to the expected number of competitors for different task categories.

The bidding system also implements a confidence-calibration mechanism based on historical outcomes: Transporters with consistent successes in specific task classes gradually increase bid aggressiveness for those classes, while Transporters that experience failures adopt more conservative bidding until their performance improves. This dynamic is formalised in Algorithm 4.16.

Algorithm 4.16 Evolutionary Bidding Strategy

1: **procedure** GenerateBid($rover$, $task$)
2: $distance \leftarrow$ computeDistance(rover.position, task.position)
3: $capacity \leftarrow rover.capacity$
4: $success\_history \leftarrow rover.success\_history$
5: $exploration\_factor \leftarrow$ random(0,1)
6: $fitness \leftarrow \beta \cdot \tilde{distance} + \gamma \cdot \tilde{capacity} + \delta \cdot \tilde{success\_history} + \epsilon \cdot exploration\_factor$
7: **return** $fitness$
8: **end procedure**

Individual policies are continuously refined by a learning mechanism that adjusts the policy weights according to feedback from completed tasks. This refinement is performed with a conservative learning rate to ensure stability and to prevent abrupt behavioural oscillations that might jeopardise operational predictability.

The system also provides policy persistence: learned strategies are persisted across operational sessions so that Transporters retain and further refine their policies over multiple missions, yielding cumulative improvements in performance over time.

4.5 Environment Development

This section presents the development of a Mars use case inspired by the Perseverance traverse. The simulated area covers approximately ±2 km and is discretised into a regular grid in which each cell corresponds to the rovers' basic movement increment. We detail the grid configuration, obstacle and sample placement, and the integration of the climate model, and we justify the modelling choices and operational strategies adopted to maximise realism within the constraints and capabilities of the previously trained models.

4.5.1 Use Case: Crater Jezero

Jezero Crater on Mars is of considerable scientific interest due to its geological history and its potential to have hosted microbial life in the past. The crater has an approximate diameter of 49 km and is regarded as one of the most promising targets for astrobiological exploration on the Red Planet. The presence of an ancient lake basin and a fluvial delta indicates that liquid water once persisted in the region, thereby increasing the likelihood of preserved biosignatures.

Figure 4.16: Image satellite of Jezero Crater, Mars

A use case centred on NASA's Perseverance rover was selected; the study area corresponds to the location traversed by the vehicle at coordinates 77.43750460° longitude and 18.46892814° latitude. The sector exhibits predominantly flat relief punctuated by small impact craters and scattered rock outcrops, a combination of features that makes it particularly well suited for realistic exploration and terrain-development simulations. The selected region is illustrated in Figure 4.16.

4.5.2 Grid Configuration and Map Editor

For the use case simulation a 200×200 grid was implemented, where each cell represents approximately 11 m in real distance, based on scale estimates derived from Perseverance rover data. Consequently, the grid covers roughly 5 km^2, an extent suitable for simulating exploration of a compact area while preserving substantial geological diversity.

Each cell is modelled as an object encapsulating meteorological attributes (ground temperature, air temperature, relative humidity, wind speed, and solar radiation). These values are generated by the Martian climate model and vary with the cell's position in the grid: cells located further from the centre receive larger percentage perturbations to their meteorological variables, producing a dynamic environment and avoiding static behaviour in the simulation.

To facilitate scenario creation and modification, an interactive editor was developed that loads the 200×200 grid together with the reference image for the selected use case. The interface enables users to place obstacles (e.g. craters, rocks, geological formations) and samples strategically, and to adjust predefined configuration parameters. On completion the editor exports a JSON file containing a dictionary of obstacle and sample locations on the grid; this file can be reloaded subsequently, allowing map revisions without reconstructing it from scratch.

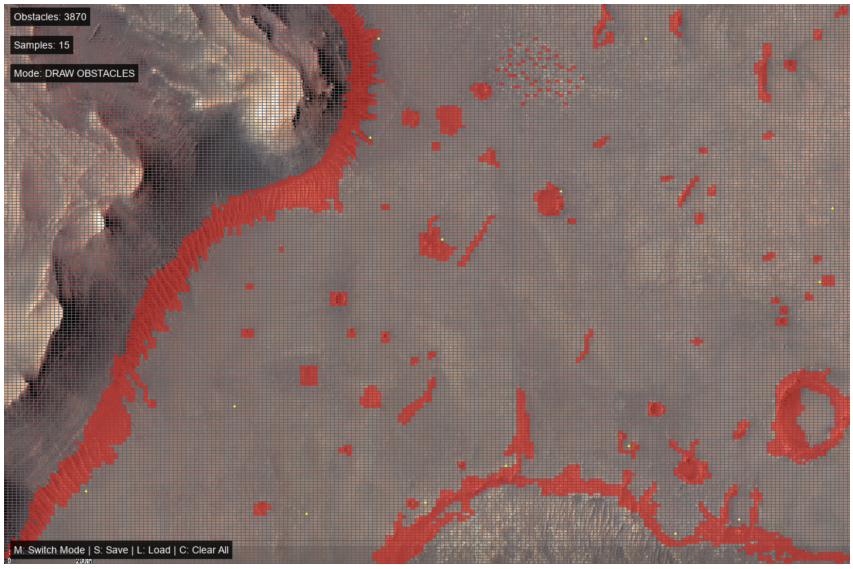

Figure 4.17: Editor interface for positioning obstacles and samples

Obstacles were introduced to emulate features that constitute operational hazards, while samples were strategically distributed to encourage traversal under adverse conditions, including regions of high obstacle density.

Climate system

An environmental climate subsystem was implemented on the grid to simulate Martian meteorological conditions using the previously trained forecasting model. This subsystem produces cell-level predictions of ground temperature, air temperature, relative humidity, wind speed and incident solar radiation. Forecasts are conditioned on local time and LS, allowing the simulated environment to reproduce realistic diurnal and seasonal variability.

The forecasting model requires a 10-hour context to generate a 5-minute horizon prediction. Accordingly, a preprocessed 10-hour input file is loaded at simulation start; the model produces the initial 5-minute prediction, which is appended to a rolling buffer containing 120 entries (equivalent to 10 hours). At each simulation step the oldest buffer entry is discarded and the newest prediction appended, thereby maintaining an updated forecast stream with 5-minute resolution that adapts to incoming inputs.

For improved situational visualisation a day/night indicator was implemented based on the RDS_T channel. This signal controls the simulation's illumination state and facilitates clear identification of dawn and dusk transitions for both display and behaviour adaptation.

Environment Simulator

An environment simulator was implemented to instantiate the map described above. The simulator comprises a visualisation area that renders the terrain, active rovers and the live simulation state, together with a control panel that aggregates operational telemetry and provides interactive controls to enable or disable specific simulation features. The simulator interface is illustrated in Figure 4.18.

Figure 4.18: Environment simulator interface

Control Panel and Visualization

An interactive control panel was implemented to allow users to interact with the simulation. The panel is organised into two tabs (VIZ and ENV). The VIZ tab provides a set of toggle buttons that enable or disable visualization features, for example:

- **Obstacles loaded from the editor**: display obstacles placed using the editor.
- **Samples loaded from the editor**: display sample locations defined in the editor.
- **Defined landing base**: show the initial landing/base position.
- **Grid overlay**: render the full map grid.
- **Radiation overlay**: visualise environmental radiation using a colour map.
- **LIDAR range**: display the sensing radius of explorer rovers.
- **Rover paths**: show planned and executed trajectories of rovers.
- **Shadow zones**: highlight shelter areas usable to reduce radiation exposure.
- **Fog-of-War mode**: reveal map cells only when rovers traverse them.

These visualization options are illustrated in Figure 4.19, which shows how each feature is represented within the simulator.

Figure 4.19: Control panel - VIZ

The second tab (ENV) contains information about the environment, such as:

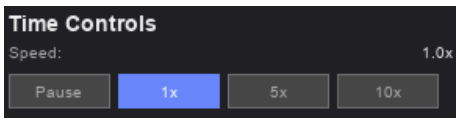

Figure 4.20: Control panel of time - ENV

A list of buttons that allows you to control the simulation time, as shown in Figure 4.20.

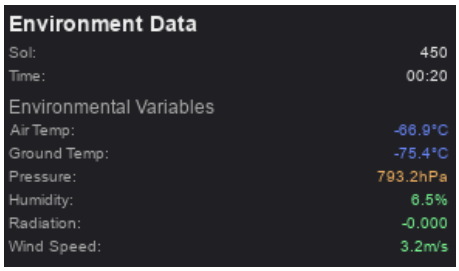

Figure 4.21: Environment Information - ENV

Directly below this control list, the "Environment Data" panel reports the current SOL and the simulation clock, which advances in five-minute increments. Within the same panel, the "Environment Variables" subsection presents the climate model's cell-level forecasts — air temperature, ground (surface) temperature, atmospheric pressure, relative humidity, incident radiation and horizontal wind speed — enabling users to monitor environmental conditions that are relevant to navigation, sensing and mission planning (see Fig. 4.21).

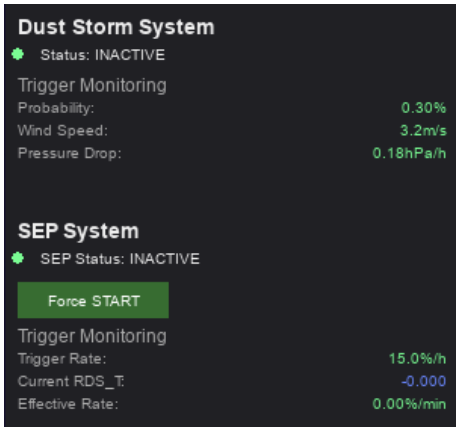

Figure 4.22: Dust storm information and SEP - ENV

Immediately below the meteorological information, the "Dust Storm and SEP" panel reports the current event status, indicating whether a dust storm or a SEP episode is active. Each indicator displays a binary state and associated diagnostic parameters to support rapid situational awareness and operational decision-making (see Fig. 4.22).

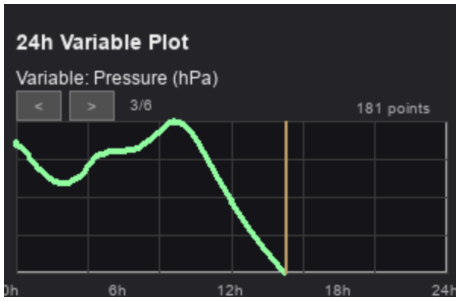

Figure 4.23: Graph showing meteorological variables at each time step - ENV

Finally, a chart presents the temporal evolution of the meteorological variables at five-minute resolution. This view enables the user to inspect diurnal patterns and short-term variability; an on-screen selector allows cycling through the six forecast variables (air temperature, ground temperature, atmospheric pressure, relative humidity, incident radiation — RDS_T — and horizontal wind speed), as illustrated in Figure 4.23.

4.5.3 Complete Simulation Pipeline

The complete simulation integrates all developed components — the Martian environment, explorer and transporter rovers, the climate subsystem, the MARL coordination layer, and the control panel. The following pseudocode summarises the full simulation pipeline, detailing the principal stages from system initialization through the main simulation loop.

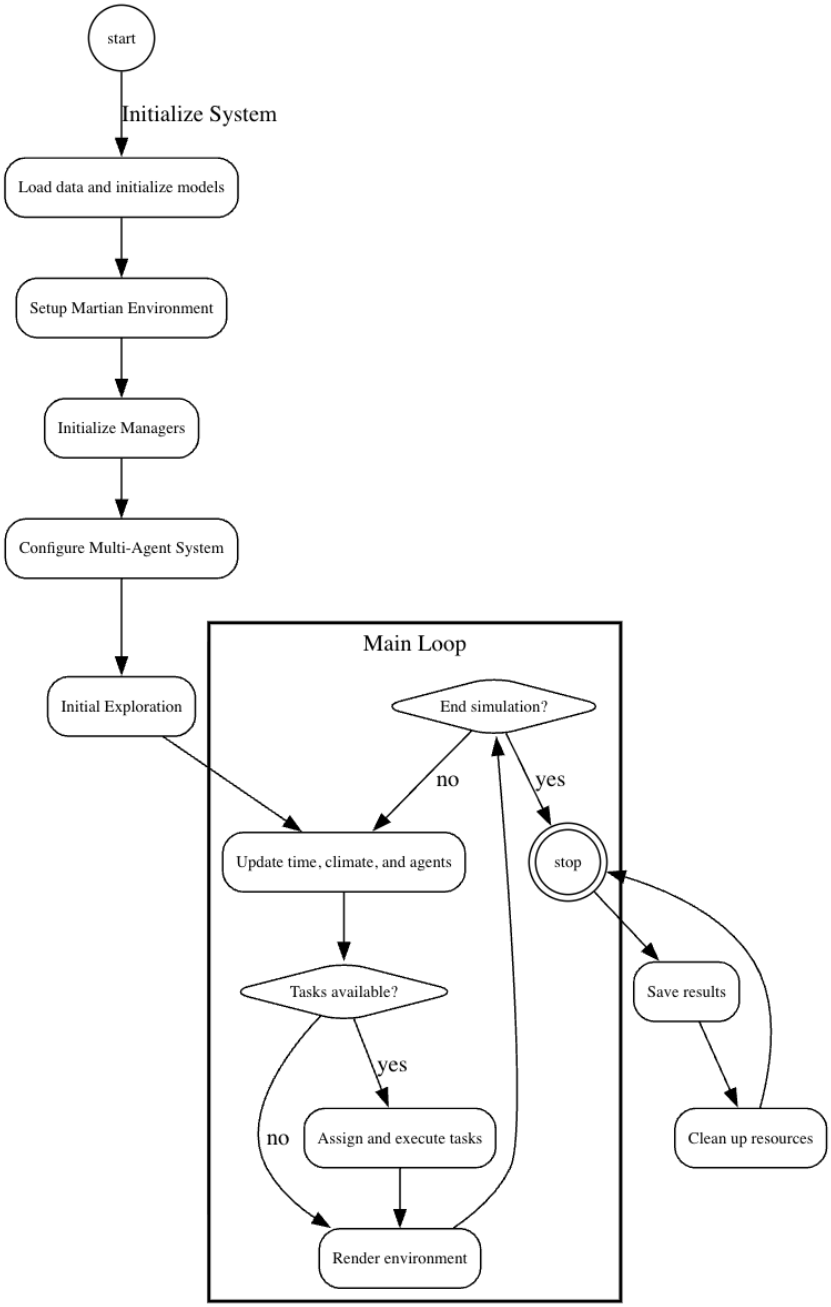

Figure 4.24: Complete Simulation Pipeline

The Martian exploration simulator is a layered system that integrates multiple technological components to reproduce, in digital form, the operational conditions and challenges of a planetary exploratory mission. Its core architecture is organised as a structured pipeline that coordinates artificial intelligence modules, environmental modelling, temporal management and graphical rendering, thereby providing a robust platform for research in collaborative robotics and autonomous planetary exploration.

System Initialization Phase

System initialization comprises a sequence of procedures that establish the computational foundations required for the simulation. The graphical subsystem is initialised (the implementation uses Pygame for real-time rendering), and historical Martian climate data are loaded from a preprocessed file to provide an empirical basis for subsequent forecasting.

The climate forecasting component is instantiated using the N-BEATS model, a neural architecture tailored for time-series prediction. A rolling historical buffer of 120 climate records is created to supply the temporal context necessary for accurate short-term forecasts. The Martian environment is realised as a two-dimensional grid of 200×200 cells, yielding 40,000 discrete positions that represent the simulated terrain.

Martian Environment Configuration

Environment modelling requires several subsystems to faithfully reproduce planetary conditions. Terrain features are loaded from the configuration file 'terrain.json', which contains predefined data for topographic obstacles, impact craters and rock formations used in this scenario. Scientific samples are also positioned according to the same configuration file, with locations and attributes chosen to represent geologically interesting materials distributed across the map. A landing base is defined to serve as the mission's origin and as the collection point for delivered samples.

The simulator implements a fog-of-war mechanism to model progressive discovery: initially only the area surrounding the landing site is known, and the remainder of the map is revealed as robotic agents explore. This behaviour reflects the epistemic limitations typically encountered in real exploratory missions.

System Managers Initialization

The simulator's modular architecture requires coordinated startup of multiple specialised managers. The temporal manager implements the Martian chronology (a Martian sol is approximately 24 h 37 min) and computes solar longitude dynamically to support seasonal modelling. The climate manager orchestrates the N-BEATS forecasting pipeline and continuously updates environmental conditions from historical data and model projections.

The dust-storm manager monitors atmospheric indicators and probabilistically simulates extreme weather events that may significantly affect operations. Event intensity is modelled using atmospheric pressure, wind speed and temperature as input variables. An automated feedback mechanism adjusts climatic parameters every five Martian sols to correct long-term drift and preserve simulation fidelity.

Multi-Agent System Configuration

The multi-agent system is one of the pipeline's most sophisticated components. The CommunicationCenter provides the local coordination infrastructure that enables near-instantaneous sharing of information among agents. It functions as a centralised repository that aggregates agent messages, maintains the shared explored map, tracks discovered samples, and coordinates task allocation via a lightweight message aggregation mechanism.

Explorer rovers are initially positioned near the landing base, while transporter rovers are configured with specialised algorithms for sample collection and delivery. All agents are

registered with the CommunicationCenter, establishing a collaborative network that supports the orchestration of complex missions.

Agent Behaviour Configuration

Each agent category is configured with role-specific behaviours. Explorers are assigned discovery-oriented policies that prioritise directions reducing overlap and maximising collective coverage. Each explorer is equipped with a virtual LIDAR sensor with a three-cell detection radius to support local mapping and obstacle detection in real time.

Transporter rovers are configured for sample handling and delivery and are registered with the MARL layer to exploit adaptive learning capabilities. When available, pretrained decision policies are loaded to accelerate behavioural optimisation. The immediate vicinity of the landing base is automatically explored within a five-cell radius to establish a foundational knowledge of local terrain for subsequent operations.

Main Simulation Loop

The simulator's operational core is the main loop, which executes continuously until termination criteria are met. Each iteration performs a sequence of update phases that maintain system coherence and synchronisation across components.

Temporal and climatic updates are performed first: the current Martian time (sol, hour, minute) is computed, solar longitude is updated, and the climate model (N-BEATS) is used to project short-term environmental conditions. Day/night state is determined from the RDS_T channel using a threshold (0.8) to distinguish between diurnal and nocturnal periods.

Radiation Management System

Radiation monitoring is a critical simulation subsystem that continuously evaluates the occurrence of solar radiation events that may compromise rover safety or sample integrity. The system tracks ambient radiation levels and enforces automatic safety protocols when predefined thresholds are exceeded. Radiation state is synchronised across subsystems to ensure a coordinated response to hazardous events.

MARL-based Task Negotiation System

Task negotiation implements a sophisticated coordination mechanism that optimises resource allocation and minimises redundant activity. Each cycle begins by pruning obsolete tasks and creating new tasks for recently discovered samples. Available transporters are identified and automatic auctions are conducted in which agents bid for tasks according to efficiency and proximity criteria.

When tasks and free transporters coincide, competitive auctions are executed using optimisation heuristics that consider distance, payload capacity and historical performance. Winners are assigned the tasks, creating contractual commitments that guide subsequent agent actions.

Explorer Rovers Update

Explorer updates encompass specialised operations that maximise exploratory effectiveness. Each explorer continuously updates local radiation readings and may alter behaviour in response to hazardous conditions. Movement decisions respect assigned preferred directions and use pathfinding algorithms that avoid obstacles while maximising coverage.

Each explorer's LIDAR subsystem is refreshed continuously to provide detailed local maps and to aid recognition of geological features. Sample detection employs pattern-recognition routines to identify scientifically relevant targets; discoveries are reported immediately to the CommunicationCenter so that the collective map and task queues are updated.

Transporter Rovers Update

Transporter behaviour adapts dynamically to the agent's operational state. During planning, transporters receive auctioned assignments and compute optimised routes using advanced pathfinding that accounts for obstacles and terrain.

In the collecting phase, transporters navigate autonomously to designated samples, employing navigation routines that compensate for drift and positional uncertainty. On arrival, transporters execute automated collection procedures that emulate the robotic operations required for Martian sample acquisition.

During delivery, transporters return to base using routes that minimise travel time and energy consumption. Successful deliveries produce MARL rewards that positively influence subsequent policy updates and improve future performance.

Verification and Reward System

Verification of task completion and the assignment of rewards constitute a fundamental mechanism in the agents' adaptive learning. The system detects completed tasks automatically by monitoring sample state and agent positions. Rewards are computed using multiple performance criteria, including completion time and execution quality.

Policy updates use reinforcement learning algorithms that incorporate observed rewards to optimise future behaviours. Over time agents develop increasingly sophisticated strategies through accumulated experience, improving the overall effectiveness of exploration operations.

Dust Storm Management

The dust-storm subsystem continuously evaluates atmospheric conditions to estimate the probability of extreme weather events. When storms are predicted or detected, the simulator applies visibility and mobility penalties to rovers, realistically reproducing the operational constraints imposed by adverse meteorological conditions.

Automatic Feedback System

An automatic feedback mechanism applies corrective adjustments to maintain long-term simulation fidelity. Every five Martian sols, at midnight (00:00), the system compares predicted and observed conditions and applies calibrated corrections to compensate for cumulative model drift and to keep the simulation aligned with empirical baselines.

Rendering and Visualization System

The rendering subsystem provides an intuitive interface for monitoring the simulation and analysing results. The environment and agents are continuously drawn to produce a clear visual representation of ongoing operations. Real-time climate information (temperature, pressure, wind speed and radiation) is presented to support situational awareness and data inspection.

Chapter 5

Results

In this chapter we present and discuss the results arising from the implementation of the proposed systems. The analysis adopts both quantitative and qualitative perspectives to provide a comprehensive assessment of the effectiveness of the developed approaches, with particular emphasis on the climate forecasting system and the MARL framework.

The discussion focuses on three principal axes: (i) performance, with respect to the accuracy, efficiency, and predictive capability of the models; (ii) robustness, understood as the systems' resilience to adverse conditions and environmental variability; and (iii) practical applicability, emphasizing the feasibility of deploying the proposed solutions in real-world robotic exploration scenarios in unknown environments.

Accordingly, this chapter not only consolidates the contributions achieved but also highlights the potential of the proposed approaches to advance the state of the art at the intersection of environmental forecasting and autonomous decision-making in robotic systems.

5.1 Climate forecasting system

In this section, we present the results obtained with the developed climate forecasting system, which integrates advanced data-imputation techniques and time-series models to predict critical meteorological variables in challenging environments, such as the Martian surface.

5.1.1 Results from Simple Imputer and KNN Imputer

With the data imputed, we can perform a more detailed inspection of the resulting time series and their behaviour. Figure 5.1 presents three scenarios: (1) the SimpleImputer prediction for SOL 62; (2) an example of a filled SOL to illustrate typical behaviour across a day–night cycle; and (3) the same SOL filled using the KNN imputer. From the comparison, the KNN imputer provides superior fidelity to the diurnal patterns of the `LOCAL_RELATIVE_HUMIDITY` feature, better preserving day–night variability.

5.1.2 Results from Random Forest and VAE

The *Random Forest* model demonstrated performance much superior to the *VAE* in the task of imputing horizontal wind speed. In tests with synthetic data, the *RF* achieved a MAE of 0.0264 and a determination coefficient (R^2) of 0.9462, indicating high precision and strong explanatory power. The *VAE*, on the other hand, presented a higher MAE (0.1376) and a negative R^2 (-0.1023), suggesting that its predictions did not adequately capture

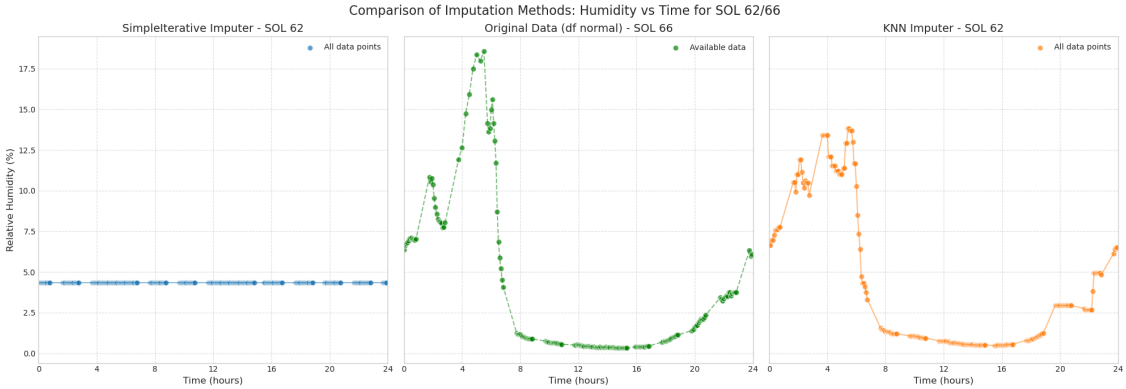

Figure 5.1: Comparison of Relative Humidity Imputation Methods

the patterns of the target variable, and were even worse than simply using the mean as a predictor.

Despite the poor standalone performance of the *VAE*, its weighted inclusion in the *ensemble* marginally contributed to the diversity of predictions, which may help to slightly smooth out noise and improve the robustness of the final model. However, the combination of models strongly favoured the *RF* (with a weight of 95%), given its much superior performance. The resulting combination maintained a low MAE (0.0271) and a high R^2 (0.9431), showing that the *ensemble* strategy did not compromise predictive quality.

The application of physical constraints (such as not allowing negative values of wind speed) and temporal smoothing with a moving window ensured that the final results were physically plausible and consistent with the expected behaviour of the Martian atmosphere. Validation with masked data reinforced the reliability of the approach adopted before applying the model to the unlabeled real data.

In summary, the *Random Forest* is clearly the dominant model in terms of performance and reliability, while the *VAE*, despite being conceptually promising, had very limited utility in this specific context. Even so, the combination of the two, when well calibrated, allowed for taking advantage of the best of each, with robustness and respect for the physical constraints of the domain.

5.1.3 Results from N-BEATS and LSTM

After training the N-BEATS and LSTM models, a performance comparison was conducted based on the MAE metric for the training and validation sets. Figure 5.2 shows the loss evolution over 100 epochs, allowing us to observe the convergence dynamics of each approach.

It is evident that N-BEATS exhibited a faster and more stable learning curve, reaching low losses early in the training and maintaining consistency between training and validation. In contrast, LSTM started with significantly higher losses, converging more slowly and stabilising at a level higher than that of N-BEATS.

At the end of the process, N-BEATS achieved superior performance, with the validation loss stabilising around 1.0, while LSTM remained at higher values, close to 5.0. These results highlight the greater efficiency and generalisation capacity of N-BEATS compared to LSTM for the analysed problem.

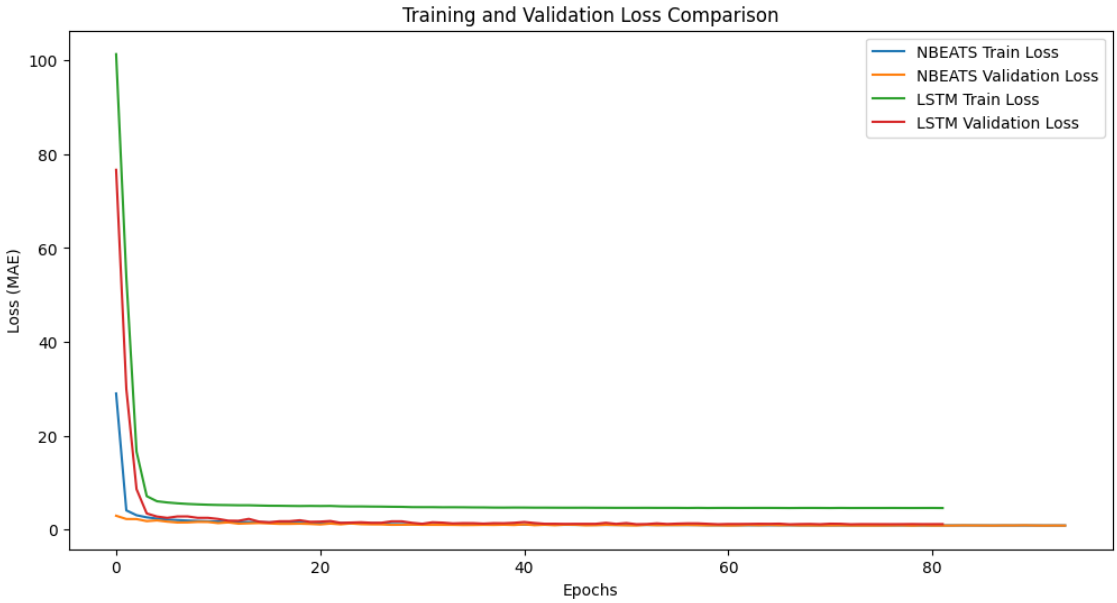

Figure 5.2: Learning Curves for N-BEATS and LSTM Models

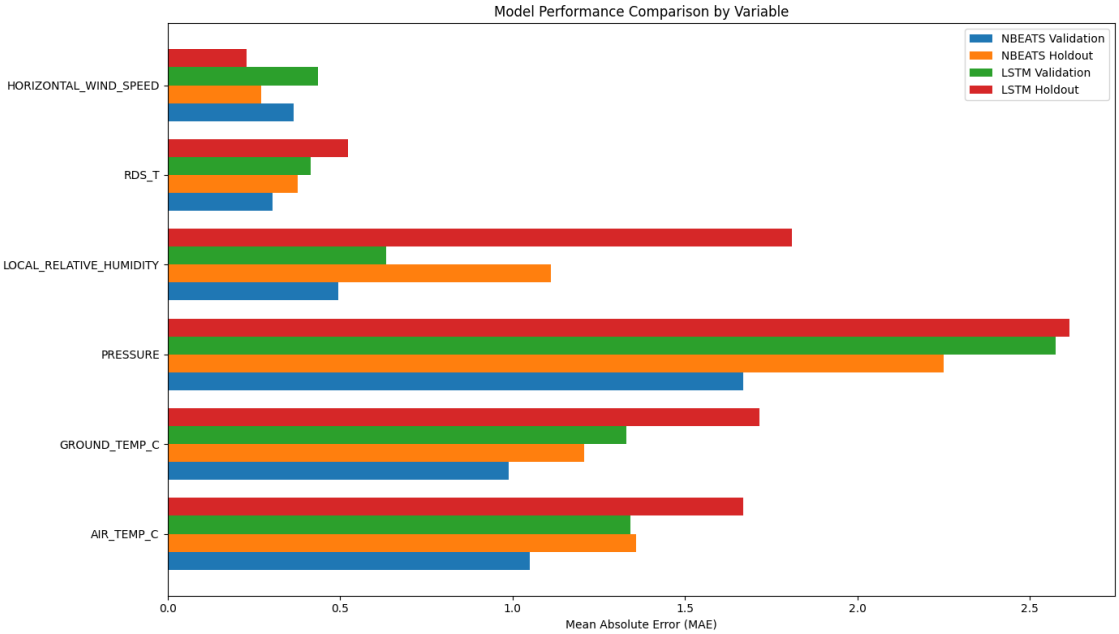

Figure 5.3: Comparison of Mean Absolute Error (MAE) by Variable and Data Set

Figure 5.3 presents a detailed comparison of performance by meteorological variable, highlighting the MAE obtained for both validation and holdout sets. Overall, the N-BEATS model demonstrated superior performance for most variables:

- For `AIR_TEMP_C`, N-BEATS achieved a MAE of approximately 1.05 (validation) and 1.25 (holdout), while LSTM showed slightly higher values, around 1.35 and 1.7, respectively.

- For GROUND_TEMP_C, both models performed similarly in validation (∼0,95 for N-BEATS and ∼1,35 for LSTM), but in the holdout set, the difference was more pronounced: 1,2 for N-BEATS compared to 1,7 for LSTM.
- The variable LOCAL_RELATIVE_HUMIDITY showed the largest discrepancy: N-BEATS maintained errors close to 0.5 (validation) and 1.2 (holdout), while LSTM ranged from 0.6 in validation to over 2.5 in holdout.
- For HORIZONTAL_WIND_SPEED, both models exhibited low errors, around 0.4–0.6 in all scenarios, with N-BEATS slightly leading in validation and LSTM in holdout.
- Finally, for the variable RDS_T, the performances were quite close, with N-BEATS obtaining about 0.3 (validation) and 0.45 (holdout), while LSTM showed values around 0.4 and 0.55, respectively.

Quantitatively, the N-BEATS model outperformed in 4 out of 5 predicted meteorological variables, especially for GROUND_TEMP_C and LOCAL_RELATIVE_HUMIDITY, where it showed considerably smaller error margins. The only case where LSTM had a slight advantage was in predicting HORIZONTAL_WIND_SPEED, although the difference compared to N-BEATS was marginal.

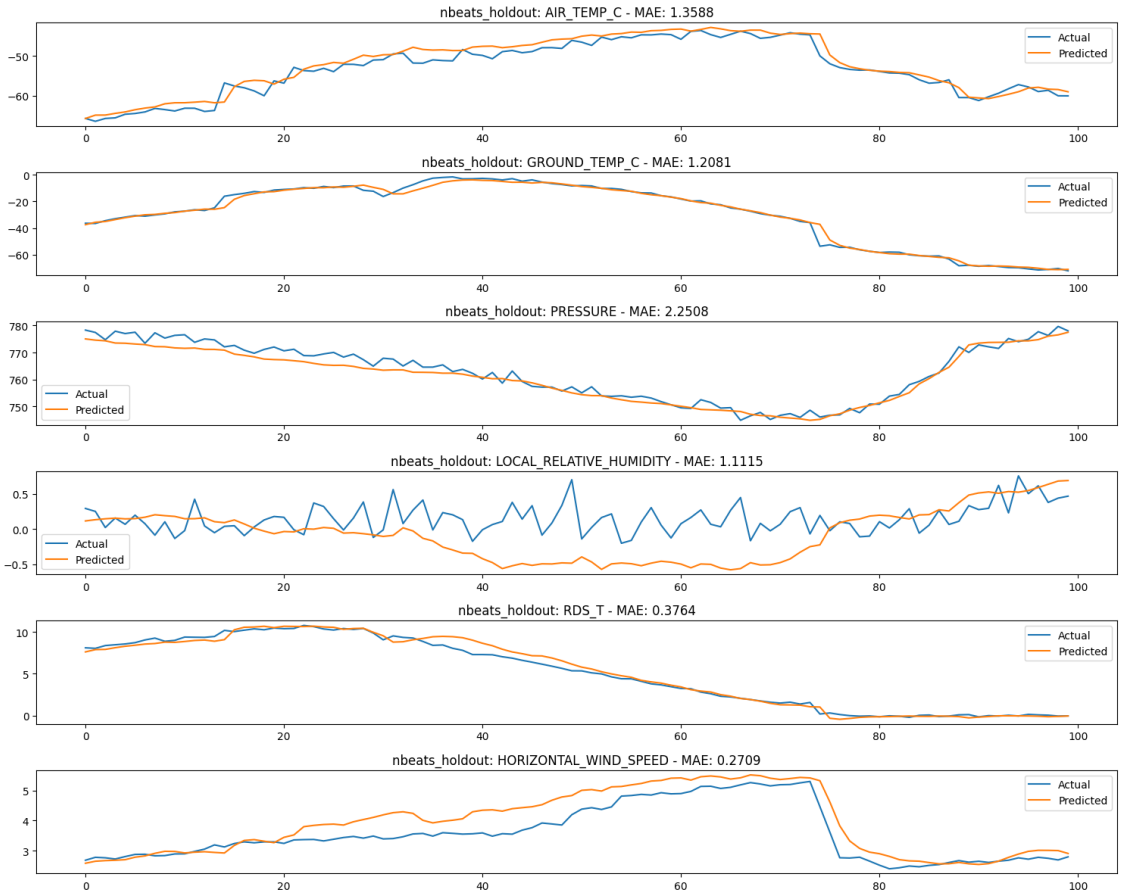

Figure 5.4: Comparison between actual and forecast values (N-BEATS - Holdout)

From a qualitative perspective, the prediction curves indicate that N-BEATS tends to produce smoother outputs that align more closely with the expected behaviour of the real

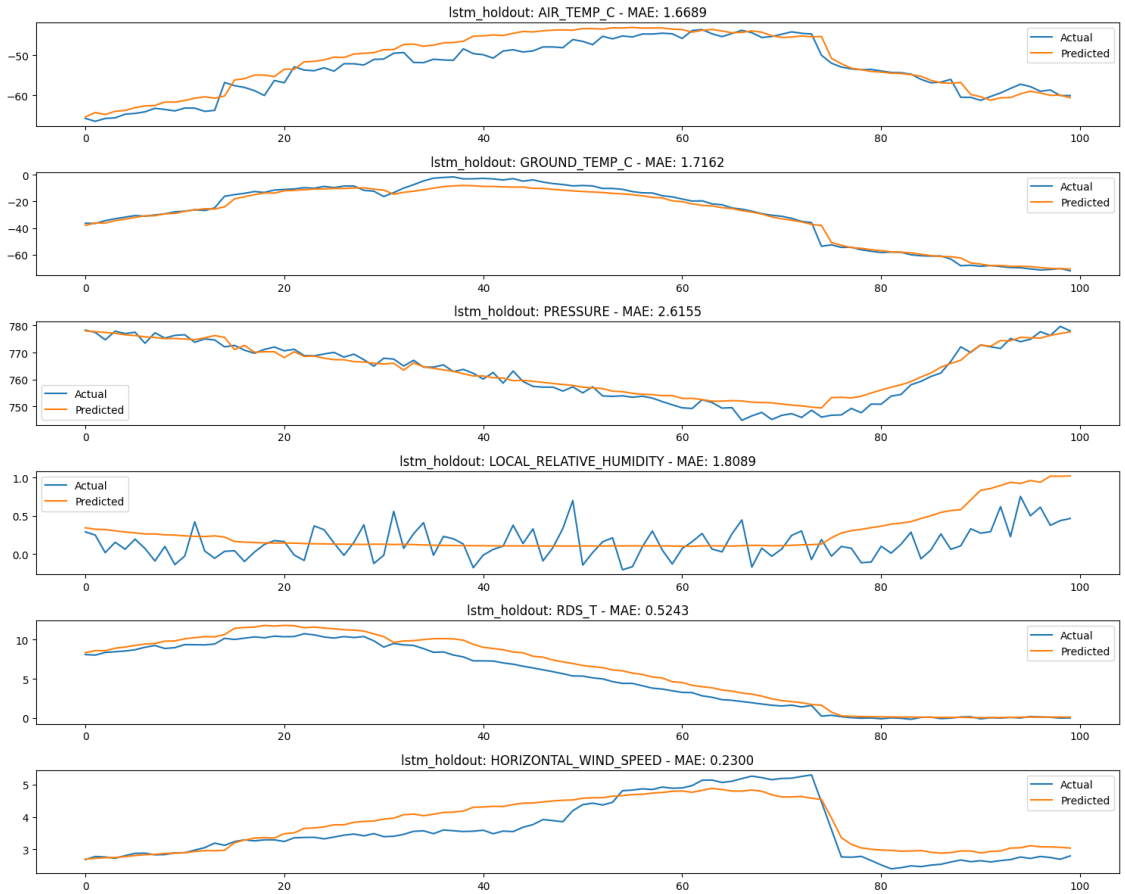

Figure 5.5: Comparison between actual and forecast values (LSTM - Hold-out)

series, particularly during periods of gradual oscillation, as illustrated in Figures 5.4 and 5.5. The residual-block structure and sequential decomposition typical of N-BEATS encourage this behaviour because the model can capture trend and seasonal components in an interpretable way. By contrast, while the LSTM is effective at modelling long-range dependencies, its forecasts exhibited greater variance and noisier behaviour—especially for wind and relative-humidity variables—likely due to LSTM's sensitivity to input noise in highly variable datasets.

Additionally, training time was comparable between the two models: N-BEATS completed adjustment in 94 epochs and LSTM in 82 epochs, given the use of strategies such as early stopping and GPU acceleration. However, N-BEATS required more memory because of its stacked-block architecture, whereas the LSTM proved computationally lighter.

In summary, the results indicate that N-BEATS was more effective for the multivariate Mars weather forecasting task, combining consistent quantitative performance with smoother and more stable visual forecasts. LSTM remained competitive on specific variables, reinforcing its suitability as a complementary approach. Model selection can thus be guided by mission-specific requirements and resource constraints.

5.2 MARL System

The system developed includes a MARL module that enables autonomous coordination among transporter agents via individualized decision policies. This section presents a quantitative analysis of the results obtained from executing the implemented system.

5.2.1 MARL System Architecture

The implemented MARL system follows a decentralized design in which each agent maintains an individual decision policy, allowing behavioral specialization while supporting global task coordination. The architecture comprises the following core components:

- **Individual Policies**: Each agent i maintains a policy $\pi_i(\theta_i)$ parameterized by weights $\theta_i = \{w_{\text{dist}}, w_{\text{load}}, w_{\text{conf}}, w_{\text{eff}}, w_{\text{risk}}\}$;
- **Auction Mechanism**: A bid-based coordination mechanism for task allocation;
- **Reinforcement Learning**: Policy updates driven by rewards obtained from successful task execution;
- **State Persistence**: Policy storage to preserve learning across sessions.

5.2.2 System Performance Metrics

The quantitative evaluation of the MARL system was conducted on real execution data collected during September 2025. The results demonstrate the effectiveness of the implemented approach.

Coordination and Learning

The system was evaluated with $N = 2$ transporter agents operating across multiple simulation sessions. Key metrics are shown in Table 5.1.

Table 5.1: Performance Metrics of the MARL System

Metric	Value
Trained agents	2
Total coordination decisions	63
Rewards assigned	50
Successful coordination rate	79.4%
Mean reward per agent	0.794

The successful coordination rate of 79.4% indicates a substantial effectiveness of the auction-based allocation mechanism in assigning and executing tasks among autonomous agents.

Policy Diversification

The analysis of behavioral differentiation among agents reveals significant specialization in the learned policies. Figure 5.6 presents a comparison of policy weights between the trained agents.

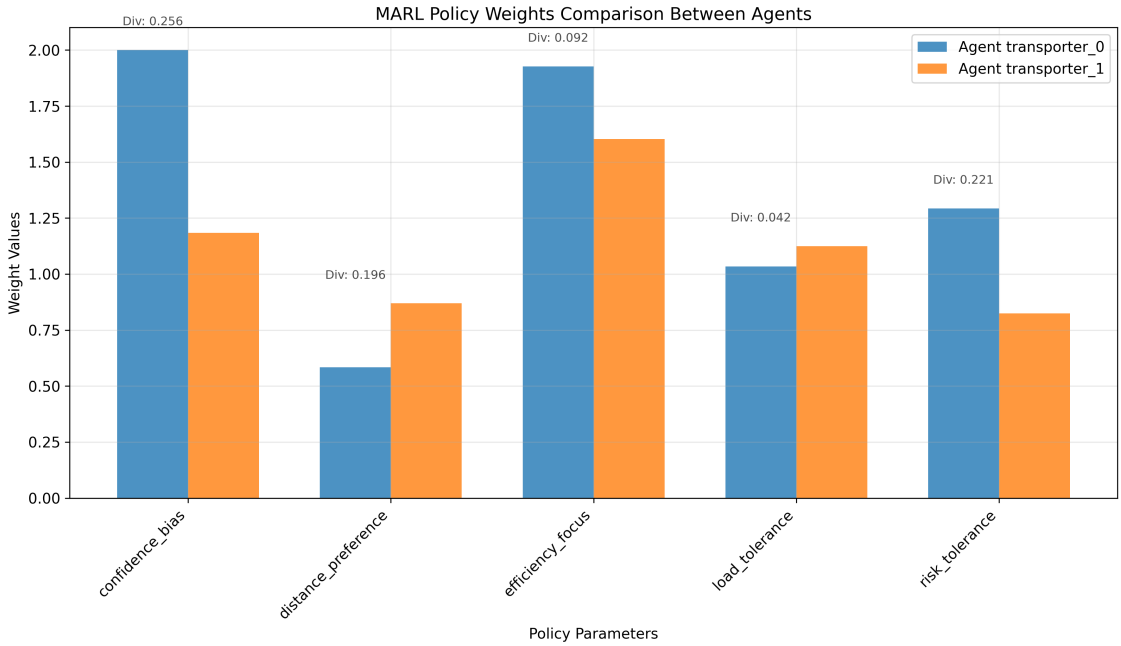

Figure 5.6: Comparison of policy weights across MARL agents. Parameter diversity indicates adaptive behavioral specialization.

Policy diversity was quantified using the coefficient of variation $CV = \frac{\sigma}{\mu}$ for each policy parameter. The results, shown in Table 5.2, demonstrate notable differentiation, particularly for the `confidence_bias` parameter.

Table 5.2: MARL Policy Diversity Analysis

Policy Parameter	Diversity (CV)	Range	Interpretation
`confidence_bias`	0.256	[1.18, 2.00]	High specialization
`risk_tolerance`	0.221	[0.83, 1.29]	Moderate differentiation
`distance_preference`	0.196	[0.58, 0.87]	Distinct spatial strategies
`efficiency_focus`	0.092	[1.60, 1.93]	Low variation
`load_tolerance`	0.042	[1.03, 1.12]	Homogeneous behavior

5.2.3 Learning Performance Analysis

Individual agent performance during the learning process is illustrated in Figure 5.7, which shows the distribution of decisions and rewards among the trained agents.

Agent `transporter_0` exhibited higher decision activity (35 decisions) compared to `transporter_1` (28 decisions), while maintaining comparable success rates. This disparity in activity suggests emergent specialization driven by the specific task distributions encountered during training.

5.2.4 Scalability Analysis

To evaluate the benefits of the multi-agent approach, a scalability study was performed comparing configurations with different numbers of explorer agents. The results are presented

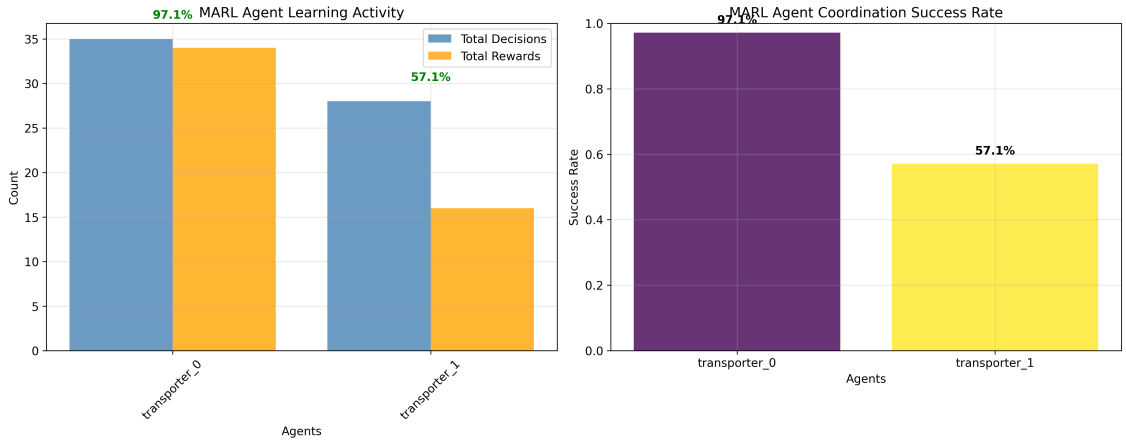

Figure 5.7: Individual learning performance of MARL agents. (a) Comparison between decisions made and rewards obtained. (b) Coordination success rate per agent.

in Figure 5.8.

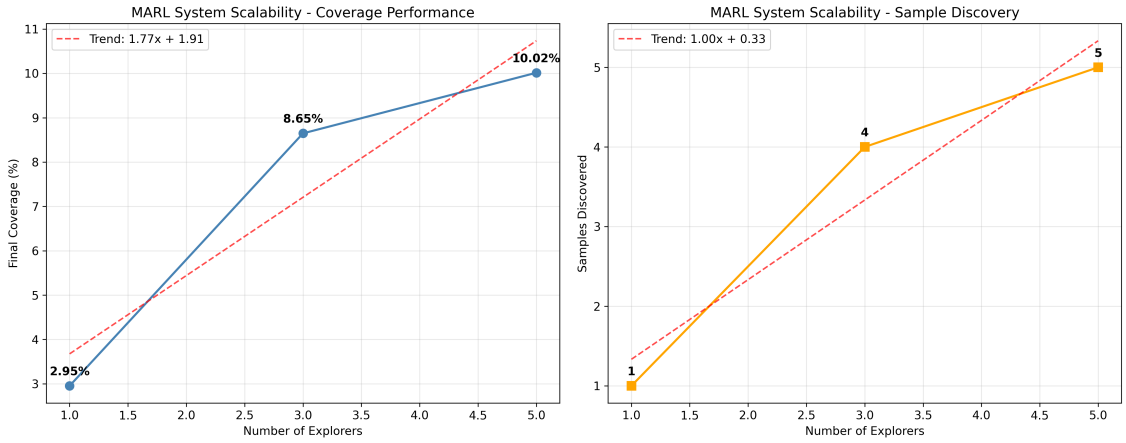

Figure 5.8: Scalability analysis of the multi-agent system. (a) Territorial coverage evolution as the number of explorers increases. (b) Correlation between number of agents and samples discovered.

The results indicate a positive linear correlation between the number of agents and system performance:

$$extCoverage(\%) = 2.535 \times N_{\text{agents}} + 0.415 \tag{5.1}$$

$$extSamples = 1.0 \times N_{\text{agents}} \tag{5.2}$$

where N_{agents} denotes the number of explorer agents in the system.

5.2.5 Multi-Agent Efficacy

The comparison between single-agent and multi-agent configurations reveals substantial benefits of coordinated operation. The configuration with three explorer agents achieved a 192.9% improvement in territorial coverage compared to the single-agent configuration (2.95% vs. 8.65%).

Table 5.3: Single-Agent vs Multi-Agent Performance Comparison

Configuration	Final Coverage (%)	Samples Discovered	Improvement (%)
1 Explorer	2.95	1.0	—
3 Explorers	8.65	4.0	+192.9
5 Explorers	10.02	5.0	+239.7

5.2.6 System Overview

Figure 5.9 provides an integrated view of the MARL system metrics, including system maturity, policy diversity, temporal evolution of coverage and the quantified benefits of multi-agent coordination.

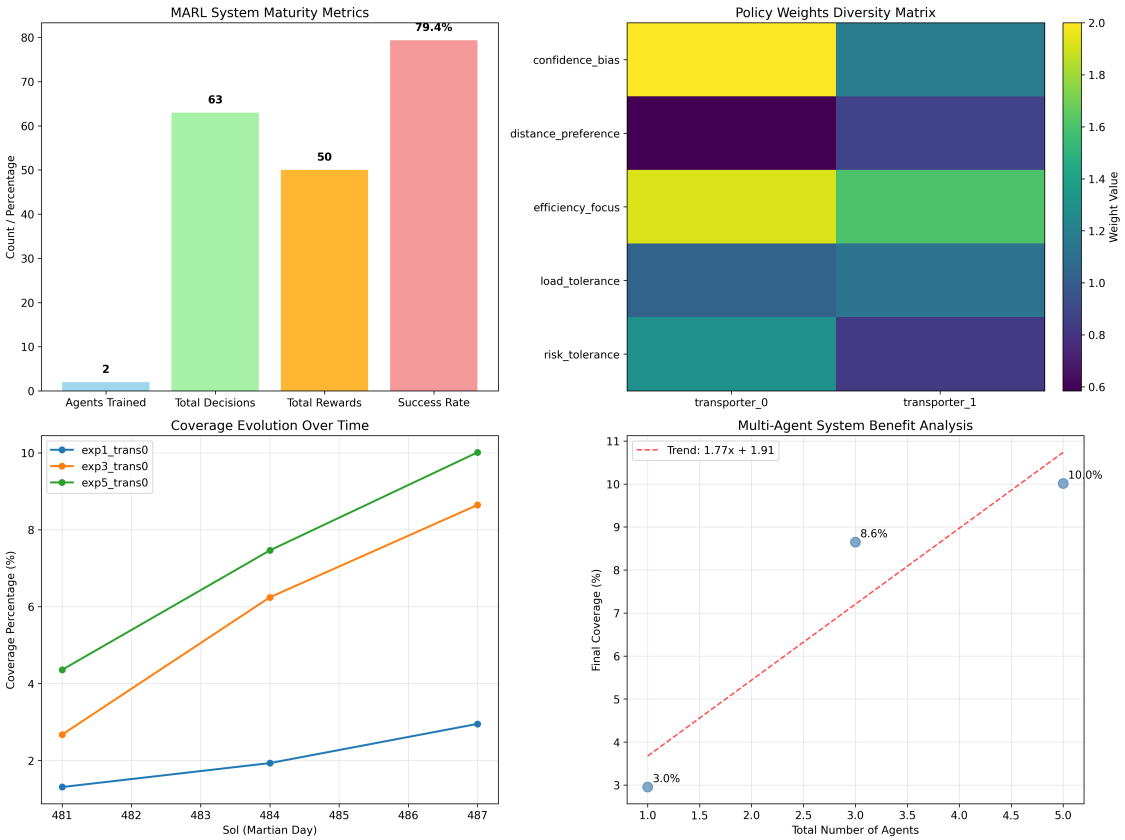

Figure 5.9: Integrated overview of the implemented MARL system. (a) System maturity metrics. (b) Policy weight diversity matrix. (c) Temporal evolution of coverage for different configurations. (d) Analysis of multi-agent benefit.

5.3 Discussion of Results

At the level of climate forecasting, the experimental results demonstrate that the system was able to capture several of the expected diurnal and seasonal patterns, particularly with respect to air and ground temperature variables. Furthermore, the model reproduced correlations typically observed during episodes of abrupt decreases in atmospheric pressure, as well as the secondary effects of these events on other measurements. Nonetheless, a limitation was identified in relation to missing observations on specific sols. This data sparsity introduced bias into the model and, in some instances, caused its predictive behaviour to approximate that of a simple autoregressive mechanism. Despite this limitation, the model proved capable of identifying meaningful patterns and of producing forecasts with a reasonable degree of accuracy, thereby establishing a robust foundation for subsequent research.

Beyond the climate-forecasting component, the results also corroborate the effectiveness of the proposed MARL implementation. The system achieved a coordination rate of 79.4%, which evidences that the auction-based allocation mechanism enabled efficient task assignment among autonomous agents. Moreover, the policy-diversity analysis revealed significant behavioural differentiation, particularly with respect to the `confidence_bias` parameter (CV = 0.256), suggesting the emergence of complementary strategies and a process of functional specialization among the agents. Another relevant aspect concerns the scalability of the architecture: the linear relationship observed between the number of agents and system performance (Equations 5.1 and 5.2) indicates that the approach can be expanded to larger multi-agent configurations without loss of efficiency. Finally, the experiments quantified the benefits of the multi-agent configuration, revealing a 192.9% improvement in territorial coverage when compared to single-agent baselines, thereby reinforcing the advantages of coordinated strategies in complex environments.

Taken together, these findings are encouraging and point towards multiple promising research directions. It is important to emphasize, however, that the present analysis constitutes an initial evaluation, conducted with only two transporter agents and over a limited timeframe. As such, the obtained policies have not yet fully converged, which highlights the need for further refinement.

Future work should therefore focus on evaluating the generalization capacity of the proposed architecture in environments with different characteristics, as well as on exploring the effects of larger agent populations and longer training horizons.

Chapter 6

Conclusions

In this chapter, an overview of the study developed, and the objectives achieved are presented. Following the study results, the limitations encountered, and future work suggestions are presented.

6.1 Summary and Objectives Achieved

This study proposed the development of a simulation framework that leverages MARL to advance autonomous robotic exploration in uncertain environments. The approach emphasized adaptive coordination among specialized agents within dynamic settings modeled from real planetary data. By integrating neural forecasting techniques to anticipate environmental variability, the project sought to strengthen system robustness and address the critical challenges of autonomy under communication delays.

The present study investigates the central research question: To what extent does the integration of MARL algorithms with dynamic environmental simulations enhance the efficiency and autonomy of robotic exploration in uncharted terrains, such as the Martian surface? To address this question, a simulation framework was developed that integrates MARL with climate forecasting models constructed from planetary datasets. The framework was designed to approximate realistic environmental variability, thereby allowing robotic agents to adapt their coordination strategies under conditions intended to emulate the uncertainties likely to be encountered during extraterrestrial exploration.

The results indicate that, while the simulation is not fully realistic, it already incorporates a range of physical constraints and environmental dynamics inspired by the Martian context. This approximation to reality is an important step because it enables evaluation of agent performance in scenarios that respect plausible physical limitations.

With respect to the climate-forecasting system, the application of data-imputation techniques successfully mitigated gaps in the records and preserved the coherence of diurnal variability. Model comparison showed that N-BEATS stood out for its consistency and robustness, capturing seasonal patterns and trends more effectively than the alternatives tested. Although models such as LSTM and VAE proved useful in specific contexts, N-BEATS was better suited for multivariate forecasting in highly variable environments. Consequently, the meteorological forecasts contributed to the resilience of autonomous exploration by providing agents with advance information about critical environmental conditions.

The implemented MARL module demonstrated that efficient coordination among agents can be achieved through decentralized policies and auction-based task-allocation mechanisms. A near-80% success rate confirms the viability and effectiveness of the negotiation process.

Policy analysis further revealed emergent specialization: agents developed distinct and complementary strategies, adapting their behaviour to contextual demands. Another notable outcome was the system's linear scalability, which showed that adding agents yields proportional improvements in territorial coverage and sample discovery. Compared to single-agent configurations, multi-agent setups produced substantially greater coverage gains—exceeding 190% in the scenarios evaluated.

Accordingly, the primary objectives set at the outset of this study have been broadly met. We developed a climate-forecasting system adapted to Mars-like conditions, implemented a multi-agent coordination module capable of generating specialized and complementary behaviours, and demonstrated the added value of collective approaches over isolated solutions. Crucially, these two components were integrated within a single simulation environment, showing that environmental forecasting can be effectively combined with multi-agent learning to enhance the robustness and autonomy of robotic exploration systems.

Despite these advances, several limitations should be noted. The simulation does not yet capture the full complexity of the Martian environment—neither in terms of climatic granularity nor with respect to extreme or unanticipated events. Similarly, agent learning was evaluated over relatively short time horizons, which does not guarantee full generalization to longer or more adverse scenarios. These limitations, however, point to clear avenues for future work, including the incorporation of higher-resolution climatic datasets, the integration of additional virtual sensors, and experimentation with more sophisticated algorithms, including quantum-enhanced variants of evolutionary and reinforcement-learning methods.

In summary, the findings indicate that integrating MARL with dynamic environmental simulations significantly improves the efficiency and autonomy of robotic exploration in unknown terrains such as the Martian surface. By combining these approaches, agents can anticipate critical environmental conditions via forecasting, autonomously adjust individual policies based on context, and cooperate in a coordinated manner to maximize territorial coverage and sample discovery. This process yields more efficient exploration with reduced redundancy among agents and greater adaptability to environmental variability and uncertainty, demonstrating that coupling MARL with realistic simulations is a decisive step toward autonomous missions in unknown terrains.

Beyond the results obtained, advances in quantum computing present potentially disruptive opportunities in this domain. Quantum algorithms applied to multi-agent learning and optimization could enable simultaneous exploration of many candidate solutions with efficiency orders of magnitude higher than classical methods, dramatically reducing the time required for policy convergence and inter-agent coordination. In the context of autonomous exploration on Mars, quantum approaches could not only accelerate adaptation to dynamic and unpredictable environments but also allow the incorporation of more complex and detailed climate models without prohibitive computational cost. Thus, quantum computing emerges as a promising avenue to amplify the benefits of integrating MARL with environmental simulations, opening new perspectives for autonomous robotic missions with levels of efficiency, robustness and autonomy that are currently unattainable.

6.2 Limitations and Future Work

Despite the progress achieved in this work, there remain methodological constraints and development opportunities that deserve discussion. This section first summarizes the main

limitations identified and then outlines potential directions for future research that could help address these challenges and extend the scope of the proposal.

6.2.1 Limitations

This study has several limitations that should be taken into account when interpreting the results.

First, the simulation does not fully capture the complexity of the Martian environment. The N-BEATS model used for forecasting environmental variables exhibits a high dependence on certain rapidly varying inputs. Such sensitivity can undermine forecast reliability whenever those inputs are poorly estimated.

Second, the two-dimensional simulated environment represents another important limitation. Although it enables a plausible assessment of agent performance under physical constraints, it does not faithfully represent the three-dimensional complexity of Martian terrain, particularly with respect to spatial perception and navigation.

Third, parts of the agent system remain rule-based (for example, navigation and sample detection). The absence of a fully reinforcement-learning-based architecture limits agent adaptability and robustness in highly dynamic environments.

Finally, the evaluation was conducted with a small number of agents and over relatively short time horizons. These restrictions hinder the generalization of results to larger and more complex scenarios, as would be expected in real exploration missions.

6.2.2 Future Work

In light of the limitations above, several lines of future research may be pursued to enhance the proposed system.

On the forecasting side, adopting more advanced variants of N-BEATS—such as N-BEATSx, which incorporates exogenous variables—could reduce dependence on unstable inputs and increase model robustness.

Regarding the simulation environment, migrating to a three-dimensional space and adding additional virtual sensors and more realistic obstacles would enable a more faithful evaluation of navigation and agent coordination.

With respect to the multi-agent architecture, progressively replacing rule-based modules with reinforcement-learning components could foster greater autonomy, scalability, and adaptability. In parallel, conducting simulations with larger numbers of agents and extended durations will permit a more thorough investigation of scalability and system performance.

Finally, integrating more sophisticated methods—including quantum variants of evolutionary and reinforcement-learning algorithms—represents a promising avenue. Although still exploratory, quantum computing has shown potential to accelerate convergence and improve coordination in multi-agent systems. Research in this direction may pave the way for future robotic exploration systems with substantially greater autonomy and resilience.

Bibliography

Alcalde, Martin et al. (Oct. 2022). "DA-SLAM: Deep active SLAM based on deep reinforcement learning". In: *2022 Latin American Robotics Symposium (LARS), 2022 Brazilian Symposium on Robotics (SBR), and 2022 Workshop on Robotics in Education (WRE)*. São Bernardo do Campo, Brazil: IEEE.

Ali, Asad et al. (Mar. 2023). "Exploration of unknown environment using deep reinforcement learning". In: *2023 International Conference on Robotics and Automation in Industry (ICRAI)*. Peshawar, Pakistan: IEEE.

Baker, Bowen et al. (2022). "Emergent complexity via multi-agent competition". In: *Nature Communications* 13, p. 2302.

Botteghi, N et al. (June 2021). "Curiosity-driven reinforcement learning agent for mapping unknown indoor environments". en. In: *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* V-1-2021, pp. 129–136.

Busoniu, Lucian, Robert Babuska, and Bart De Schutter (2008). "A comprehensive survey of multi-agent reinforcement learning". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38.2, pp. 156–172.

Calzolari, Gabriele et al. (Oct. 2024). "Decentralized multi-agent reinforcement learning exploration with inter-agent communication-based action space". In: *2024 24th International Conference on Control, Automation and Systems (ICCAS)*. Jeju, Korea, Republic of: IEEE, pp. 319–324.

Cantor, Bruce A., Philip B. James, and Wendy M. Calvin (2010). "MARCI and MOC observations of the atmosphere and surface cap in the north polar region of Mars". In: *Icarus* 208.1, pp. 61–81. issn: 0019-1035. doi: https://doi.org/10.1016/j.icarus.2010.01.032. url: https://www.sciencedirect.com/science/article/pii/S0019103510000515.

Cao, Xiao et al. (June 2024). "HMA-SAR: Multi-agent search and rescue for unknown located dynamic targets in completely unknown environments". In: *IEEE Robot. Autom. Lett.* 9.6, pp. 5567–5574.

Chen, Kuan-Cheng et al. (2024). "Quantum-Train-Based Distributed Multi-Agent Reinforcement Learning". In: eprint: 2412.08845 (quant-ph).

Chen, Qihong et al. (Aug. 2024). "Transformer-based reinforcement learning for multi-robot autonomous exploration". en. In: *Sensors (Basel)* 24.16, p. 5083.

Cho, Yeryeong et al. (Oct. 2024). "Introduction to Quantum Multi-Agent Reinforcement Learning: Concepts and Applications". In: *2024 15th International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1151–1153. doi: 10.1109/ICTC62082.2024.10826726. url: https://ieeexplore.ieee.org/document/10826726/.

Colby, Mitchell, Logan Yliniemi, and Kagan Tumer (Apr. 2016). "Autonomous Multiagent Space Exploration with High-Level Human Feedback". In: *https://doi.org/10.2514/1.I010379* 13 (8), pp. 301–315. issn: 23273097. doi: 10.2514/1.I010379. url: https://arc.aiaa.org/doi/10.2514/1.I010379.

Dai, Xin et al. (2024). "Quantum machine learning architecture search via deep reinforcement learning". In: eprint: 2407.20147 (quant-ph).

Deng, Liyuan, Wei Gong, and Li Li (Nov. 2022). "Multi-robot exploration in unknown environments via multi-agent deep reinforcement learning". In: *2022 China Automation Congress (CAC)*. Xiamen, China: IEEE.

DeRieux, Alexander and Walid Saad (2024). "EQMARL: Entangled quantum multi-agent reinforcement learning for distributed cooperation over quantum channels". In: eprint: 2405.17486 (quant-ph).

Dorigo, Marco and Thomas Stützle (2019). *Ant Colony Optimization*. MIT Press.

Foerster, Jakob et al. (2018). "Stabilizing experience replay for deep multi-agent reinforcement learning". In: *International Conference on Machine Learning (ICML)*.

Gao, Tengteng et al. (Apr. 2022). "Hybrid path planning algorithm of the mobile agent based on Q-learning". en. In: *Autom. Contr. Comput. Sci.* 56.2, pp. 130–142.

Habib, Adria Binte et al. (2025). "N-BEATS & Temporal Fusion Transformer Based Surface Temperature Prediction and Forecasting for Realizing Global Warming Trends". In: *Artificial Intelligence and Speech Technology*. Ed. by Amita Dev et al. Cham: Springer Nature Switzerland, pp. 30–41. isbn: 978-3-031-75167-7.

Hamann, Arne and Sabine Wölk (Mar. 2022). "Performance analysis of a hybrid agent for quantum-accessible reinforcement learning". In: *New J. Phys.* 24.3, p. 033044.

Han, Wanbin, Chongrong Fang, and Jianping He (Oct. 2022). "Mapless path planning of multi-robot systems in complex environments via deep reinforcement learning". In: *2022 4th International Conference on Data-driven Optimization of Complex Systems (DOCS)*. Chengdu, China: IEEE.

Hernandez-Leal, Pablo, Bilal Kartal, and Matthew E. Taylor (2019). "A survey and critique of multiagent deep reinforcement learning". In: *Autonomous Agents and Multi-Agent Systems* 33.6, pp. 750–797.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory". In: *Neural computation* 9.8, pp. 1735–1780.

Hu, Junyan et al. (Dec. 2020a). "Voronoi-Based Multi-Robot Autonomous Exploration in Unknown Environments via Deep Reinforcement Learning". In: *IEEE Transactions on Vehicular Technology* 69 (12), pp. 14413–14423. issn: 19399359. doi: 10.1109/TVT.2020.3034800.

– (Dec. 2020b). "Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning". In: *IEEE Trans. Veh. Technol.* 69.12, pp. 14413–14423.

Hu, Likun, Chunyou Wei, and Linfei Yin (Feb. 2025). "Fuzzy A quantum multi-stage Q-learning artificial potential field for path planning of mobile robots". In: *Engineering Applications of Artificial Intelligence* 141, p. 109866. issn: 0952-1976. doi: 10.1016/J.ENGAPPAI.2024.109866.

Hu, Yazhou et al. (Nov. 2021). "Quantum-enhanced reinforcement learning for control: a preliminary study". en. In: *Contr. Theory Technol.* 19.4, pp. 455–464.

Jin, Yue et al. (May 2019). "Efficient multi-agent cooperative navigation in unknown environments with interlaced deep reinforcement learning". In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brighton, United Kingdom: IEEE.

Kalra, Geetansh et al. (2024). "BAMAX: Backtrack Assisted Multi-Agent Exploration using Reinforcement Learning". In: eprint: 2411.08400 (cs.RO).

Karami, Mohammad Hasan, Hossein Aghababa, and Amir Hosein Keyhanipour (Oct. 2020). "An entanglement-inspired action selection and knowledge sharing scheme for cooperative multi-agent Q-learning algorithm used in robot navigation". In: *2020 10th International Conference on Computer and Knowledge Engineering (ICCKE)*. Mashhad, Iran: IEEE.

Kölle, Michael et al. (2024). "Optimizing variational quantum circuits using metaheuristic strategies in Reinforcement Learning". In: eprint: 2408.01187 (quant-ph).

Li, Juncheng et al. (July 2021). "A behavior-based mobile robot navigation method with deep reinforcement learning". en. In: *Unmanned Syst.* 09.03, pp. 201–209.

Li, Minghao et al. (May 2022). "Decentralized global connectivity maintenance for multi-robot navigation: A reinforcement learning approach". In: *2022 International Conference on Robotics and Automation (ICRA)*. Philadelphia, PA, USA: IEEE.

Li, Xiaofeng, Jie Ren, and Yunbo Li (Jan. 2024). "Multi-mode filter target tracking method for mobile robot using multi-agent reinforcement learning". en. In: *Eng. Appl. Artif. Intell.* 127.107398, p. 107398.

Liu, Shuqi and Zhaoxia Wu (2019). "Efficient multi-robot exploration via multi-head attention-based cooperation strategy". In: eprint: 1911.01774 (cs.AI).

Lockwood, Owen and Mei Si (2020). "Reinforcement learning with quantum variational circuits". In: eprint: 2008.07524 (quant-ph).

Lowe, Ryan et al. (2017). "Multi-agent actor-critic for mixed cooperative-competitive environments". In: *Advances in Neural Information Processing Systems (NeurIPS)*.

Löwe, Jan Leo et al. (Apr. 2025). "Nowcasting Solar Energetic Particle Events for Mars Missions". In: *Space Weather* (4), e2025SW004372. issn: 1542-7390. doi: 10.1029/2025SW004372.

Moflic, Ioana, Vikas Garg, and Alexandru Paler (2023). "Graph Neural Network Autoencoders for Efficient Quantum Circuit Optimisation". In: eprint: 2303.03280 (quant-ph).

Mu, Xuechen et al. (2023). "Hierarchical Task Network planning for facilitating cooperative multi-agent reinforcement learning". In: eprint: 2306.08359 (cs.AI).

NASA Planetary Data System (PDS) (2020). *Dataset MEDA 2020*. NASA Planetary Data System. url: https://atmos.nmsu.edu/PDS/data/PDS4/Mars2020/mars2020_meda/.

Ning, Zepeng and Lihua Xie (June 2024). "A survey on multi-agent reinforcement learning and its application". In: *Journal of Automation and Intelligence* 3 (2), pp. 73–91. issn: 2949-8554. doi: 10.1016/J.JAI.2024.02.003.

Nweye, Kingsley et al. (2022). "Real-world challenges for multi-agent reinforcement learning in grid-interactive buildings". In: *Energy and AI* 10, p. 100202. issn: 2666-5468. doi: https://doi.org/10.1016/j.egyai.2022.100202. url: https://www.sciencedirect.com/science/article/pii/S2666546822000489.

Oliehoek, Frans A. and Christopher Amato (2016). *A concise introduction to decentralized POMDPs*. Springer.

Oreshkin, Boris N. et al. (May 2019). "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting". In: *8th International Conference on Learning Representations, ICLR 2020*. url: https://arxiv.org/pdf/1905.10437.

Raj, Ravi and Andrzej Kos (Oct. 2024). "Intelligent mobile robot navigation in unknown and complex environment using reinforcement learning technique". en. In: *Sci. Rep.* 14.1, p. 22852.

Redondo, Josefina Torres et al. (2016). "STUDY OF A WALKING ROBOT FOR MARS EXPLORATION". In: *DYNA New Technologies Journal* 3 (1), pp. 1–11. issn: 2386-8406. doi: 10.6036/NT7861. url: https://revista-dyna.com/dyna-newtech/index.php/dyna-newtech/article/view/65.

Reichstein, Markus et al. (2019). "Deep learning and process understanding for data-driven Earth system science". In: *Nature* 566.7743, pp. 195–204. doi: 10.1038/s41586-019-0912-1. url: https://doi.org/10.1038/s41586-019-0912-1.

Rodriguez-Manfredi, J. A. et al. (Apr. 2021). "The Mars Environmental Dynamics Analyzer, MEDA. A Suite of Environmental Sensors for the Mars 2020 Mission". In: *Space Science*

Reviews 2021 217:3 217 (3), pp. 1–86. issn: 1572-9672. doi: 10.1007/S11214-021-00816-9. url: https://link.springer.com/article/10.1007/s11214-021-00816-9.

Sánchez-Lavega, A. et al. (Jan. 2023). "Mars 2020 Perseverance Rover Studies of the Martian Atmosphere Over Jezero From Pressure Measurements". In: *Journal of Geophysical Research: Planets* 128 (1), e2022JE007480. issn: 2169-9100. doi: 10.1029/2022JE007480. url: /doi/pdf/10.1029/2022JE007480%20https://onlinelibrary.wiley.com/doi/abs/10.1029/2022JE007480%20https://agupubs.onlinelibrary.wiley.com/doi/10.1029/2022JE007480.

Shinde, Pramila P. and Seema Shah (July 2018). "A Review of Machine Learning and Deep Learning Applications". In: *Proceedings - 2018 4th International Conference on Computing, Communication Control and Automation, ICCUBEA 2018*. doi: 10.1109/ICCUBEA.2018.8697857.

Sygkounas, Alkis et al. (June 2022). "Multi-agent Exploration with Reinforcement Learning". In: *2022 30th Mediterranean Conference on Control and Automation (MED)*. Vouliagmeni, Greece: IEEE.

Taghavi, Mazyar (2024). "Quantum Computing and Neuromorphic Computing for Safe, Reliable, and explainable Multi-Agent Reinforcement Learning: Optimal control in autonomous robotics". In: eprint: 2408.03884 (cs.ET).

Tan, Ming (1993). "Multi-agent reinforcement learning: Independent vs. cooperative agents". In: *Proceedings of the Tenth International Conference on Machine Learning*, pp. 330–337.

Tse, Hon Tik and Ho-Fung Leung (2022). "Exploiting Semantic Epsilon Greedy exploration strategy in multi-agent reinforcement learning". In: eprint: 2201.10803 (cs.LG).

Turun Ursa (2021). *Mars Calendar (Mars Year 36)*. https://turunursa.fi/tiedostoja/eka/Mars.pdf. Calendário marciano criado por membros da Turun Ursa para o ano marciano 36, disponível sob licença Creative Commons. url: https://turunursa.fi/tiedostoja/eka/Mars.pdf.

Verma, Vandi et al. (July 2023). "Autonomous robotics is driving Perseverance rover's progress on Mars". In: *Science Robotics* 8 (80). issn: 24709476. doi: 10.1126/SCIROBOTICS.ADI3099/SUPPL_FILE/SCIROBOTICS.ADI3099_SM.PDF. url: https://www.science.org/doi/10.1126/scirobotics.adi3099.

Wang, Huiqun and Mark I. Richardson (May 2015). "The origin, evolution, and trajectory of large dust storms on Mars during Mars years 24–30 (1999–2011)". In: *Icarus* 251, pp. 112–127. issn: 0019-1035. doi: 10.1016/J.ICARUS.2013.10.033. url: https://www.sciencedirect.com/science/article/pii/S0019103513004624?casa_token=Gd88YhdterkAAAAA:CWf2F29PV38vOrM9YlbQ0vasU1e_cAOXbyEdLuGlSXGPNeTDqOdCf2PTEvp1JfzycX3bo8HZ

Wang, Jiayao et al. (Sept. 2023). "Bio-inspired decentralized multi-robot exploration in unknown environments". In: *2023 IEEE 20th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*. Toronto, ON, Canada: IEEE.

Wilson, Granville Tunnicliffe (Sept. 2016). "Time Series Analysis: Forecasting and Control, 5th Edition, by George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel and Greta M. Ljung, 2015. Published by John Wiley and Sons Inc., Hoboken, New Jersey, pp. 712. ISBN: 978-1-118-67502-1". In: *Journal of Time Series Analysis* 37 (5), pp. 709–711. issn: 0143-9782. doi: 10.1111/JTSA.12194.

Xie, Hongyu et al. (2023). "Autonomous multirobot navigation and cooperative mapping in partially unknown environments". In: *IEEE Trans. Instrum. Meas.* 72, pp. 1–12.

Yang, Yuguang, Michael A. Bevan, and Bo Li (Jan. 2020). "Efficient Navigation of Colloidal Robots in an Unknown Environment via Deep Reinforcement Learning". In: *Advanced Intelligent Systems* 2 (1), p. 1900106. issn: 2640-4567. doi: 10.1002/AISY.201900106.

url: https://onlinelibrary.wiley.com/doi/full/10.1002/aisy.201900106%20https://onlinelibrary.wiley.com/doi/abs/10.1002/aisy.201900106%20https://onlinelibrary.wiley.com/doi/10.1002/aisy.201900106.

Yliniemi, Logan, Adrian K. Agogino, and Kagan Tumer (Dec. 2014). "Multirobot Coordination for Space Exploration". In: *AI Magazine* 35 (4), pp. 61–74. issn: 2371-9621. doi: 10.1609/AIMAG.V35I4.2556. url: https://onlinelibrary.wiley.com/doi/full/10.1609/aimag.v35i4.2556%20https://onlinelibrary.wiley.com/doi/abs/10.1609/aimag.v35i4.2556%20https://onlinelibrary.wiley.com/doi/10.1609/aimag.v35i4.2556.

Yu, Wenhan and Jun Zhao (2023). "Quantum Multi-Agent Reinforcement Learning as an Emerging AI Technology: A Survey and Future Directions". In: *ICCA 2023 - 2023 5th International Conference on Computer and Applications, Proceedings*. doi: 10.1109/ICCA59364.2023.10401605.

Zhang, Hao et al. (Apr. 2022a). "H2GNN: Hierarchical-Hops Graph Neural Networks for Multi-Robot Exploration in Unknown Environments". In: *IEEE Robotics and Automation Letters* 7 (2), pp. 3435–3442. issn: 23773766. doi: 10.1109/LRA.2022.3146912.

– (Apr. 2022b). "H2GNN: Hierarchical-hops graph neural networks for multi-robot exploration in unknown environments". In: *IEEE Robot. Autom. Lett.* 7.2, pp. 3435–3442.

Zhang, Kaiqing, Zhuoran Yang, and Tamer Basar (2021). "Multi-agent reinforcement learning: A selective overview of theories and algorithms". In: *Handbook of Reinforcement Learning and Control*. Springer, pp. 321–384.

Zhou, Zhiguo et al. (Aug. 2024). "Mars Exploration: Research on Goal-Driven Hierarchical DQN Autonomous Scene Exploration Algorithm". In: *Aerospace 2024, Vol. 11, Page 692* 11 (8), p. 692. issn: 2226-4310. doi: 10.3390/AEROSPACE11080692. url: https://www.mdpi.com/2226-4310/11/8/692/htm%20https://www.mdpi.com/2226-4310/11/8/692.

Zhu, Shaohao et al. (2024). "MAexp: A generic platform for RL-based multi-agent exploration". In: eprint: 2404.12824 (cs.RO).